Few shot learning

Deep Learning (DSE316/616)

Vinod K Kurmi
Assistant Professor, DSE

Indian Institute of Science Education and Research Bhopal

Nov 3, 2022

Disclaimer

- Much of the material and slides for this lecture were borrowed from
  - Bernhard Schölkopf's MLSS 2017 lecture,
  - Tommi Jaakkola's 6.867 class,
  - CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University
  - Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class
  - Hongsheng Li's ELEG5491 class
  - Tsz-Chiu Au slides
  - Mitesh Khapra Class notes
  - Vineeth Balasubramanian Class

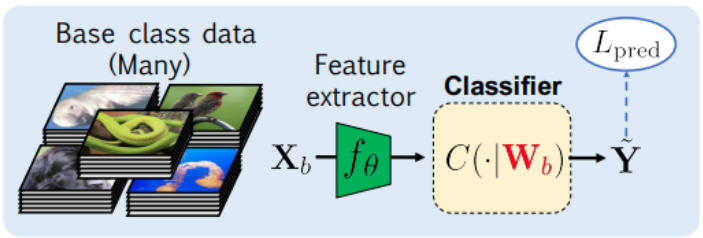
Learning with Limited Supervision

Problem

- Deep learning model ==> Heavily depends upon training ***labeled data***.
- Not directly suited to learn from few samples

Solution

- Trained model capable of rapidly generalizing to new tasks with only a few samples.
- Enable models to perform under practical scenarios.



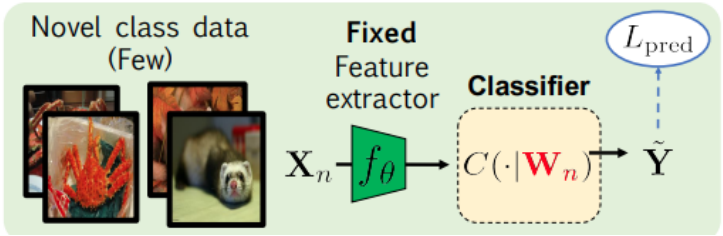


Image credit: Chen et al A Closer Look at Few-shot Classification ICLR 2019

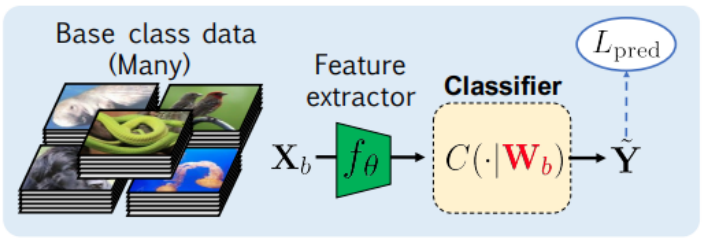
Learning with Limited Supervision

Problem

- Deep learning model ==> Heavily depends upon training ***labeled data***.
- Not directly suited to learn from few samples

Solution

- Trained model capable of rapidly generalizing to new tasks with only a few samples.
- Enable models to perform under practical scenarios.



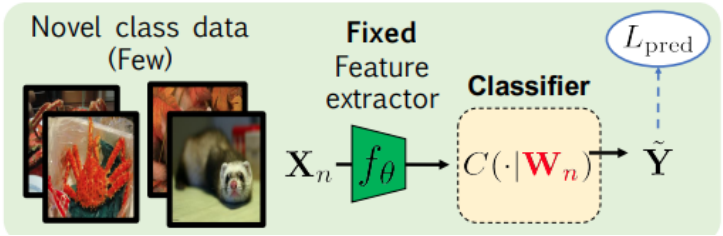


Image credit: Chen et al A Closer Look at Few-shot Classification ICLR 2019

Problem setting

Few shot Learning (FSL)

- Training Data: $D_{train} = (x_i, y_i)_{i=1}^{P}$, where few training samples for certain classes.
- **N-way-K-Shot FSL:** contains only few examples K from N overall number of classes (Other classes called base classes)

Zero shot Learning (ZSL)

- Training Data: $S = \{(x, y, a(y))|x \in X^s, y =\in Y^S, a(y) \in A\}$, where X_s is the set of image/features from seen classes, Y_s is the set of seen labeles, $a(y)$ is semantic embedding for class y.
- Test set $U = \{(x, y, a(y))|x \in X^U, y =\in Y^U, a(y) \in A\}$, where X_u is the set of image/features from unseen classes, Y_s is the set of unseen labeles, $Y^u \cap Y^S = \emptyset$.

Problem setting

Few shot Learning (FSL)

- Training Data: $D_{train} = (x_i, y_i)_{i=1}^{P}$, where few training samples for certain classes.
- **N-way-K-Shot FSL:** contains only few examples K from N overall number of classes (Other classes called base classes)

Zero shot Learning (ZSL)

- Training Data: $S = \{(x, y, a(y))|x \in X^s, y =\in Y^S, a(y) \in A\}$, where X_s is the set of image/features from seen classes, Y_s is the set of seen labeles, $a(y)$ is semantic embedding for class y.
- Test set $U = \{(x, y, a(y))|x \in X^U, y =\in Y^U, a(y) \in A\}$, where X_u is the set of image/features from unseen classes, Y_s is the set of unseen labeles, $Y^u \cap Y^S = \emptyset$.

Problem setting

Few shot Learning (FSL)

- Training Data: $D_{train} = (x_i, y_i)_{i=1}^{P}$, where few training samples for certain classes.
- **N-way-K-Shot FSL:** contains only few examples K from N overall number of classes (Other classes called base classes)

Zero shot Learning (ZSL)

- Training Data: $S = \{(x, y, a(y))|x \in X^s, y =\in Y^S, a(y) \in A\}$, where X_s is the set of image/features from seen classes, Y_s is the set of seen labeles, $a(y)$ is semantic embedding for class y.
- Test set $U = \{(x, y, a(y))|x \in X^U, y =\in Y^U, a(y) \in A\}$, where X_u is the set of image/features from unseen classes, Y_s is the set of unseen labeles, $Y^u \cap Y^S = \emptyset$.

Problem setting

Few shot Learning (FSL)

- Training Data: $D_{train} = (x_i, y_i)_{i=1}^{P}$, where few training samples for certain classes.
- **N-way-K-Shot FSL:** contains only few examples K from N overall number of classes (Other classes called base classes)

Zero shot Learning (ZSL)

- Training Data: $S = \{(x, y, a(y))|x \in X^s, y =\in Y^S, a(y) \in A\}$, where X_s is the set of image/features from seen classes, Y_s is the set of seen labeles, $a(y)$ is semantic embedding for class y.
- Test set $U = \{(x, y, a(y))|x \in X^U, y =\in Y^U, a(y) \in A\}$, where X_u is the set of image/features from unseen classes, Y_s is the set of unseen labeles, $Y^u \cap Y^S = \emptyset$.

Problem setting

Based on the classes that a model can sees in test phase, FSL/ZSL problems can be categorized into tow settings:

Conventional ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U$
- Data x to be recognized at test time belongs only to unseen/few-shot classes.

Generalized ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U cup Y^S$
- Data x to be recognized at test time belongs to seen/base or unseen/few-shot classes.

Problem setting

Based on the classes that a model can sees in test phase, FSL/ZSL problems can be categorized into tow settings:

Conventional ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U$
- Data x to be recognized at test time belongs only to unseen/few-shot classes.

Generalized ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U cup Y^S$
- Data x to be recognized at test time belongs to seen/base or unseen/few-shot classes.

Problem setting

Based on the classes that a model can sees in test phase, FSL/ZSL problems can be categorized into tow settings:

Conventional ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U$
- Data x to be recognized at test time belongs only to unseen/few-shot classes.

Generalized ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U cup Y^S$
- Data x to be recognized at test time belongs to seen/base or unseen/few-shot classes.

Problem setting

Based on the classes that a model can sees in test phase, FSL/ZSL problems can be categorized into tow settings:

Conventional ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U$
- Data x to be recognized at test time belongs only to unseen/few-shot classes.

Generalized ZSL/FSL

- Goal to learn a classifier $f : x \rightarrow Y^U cup Y^S$
- Data x to be recognized at test time belongs to seen/base or unseen/few-shot classes.

Recall Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = argmin_h R(h)$ be function that minimized the expected risk
- $h^* = argmin_{h \in H} R(h)$ be function in H that minimized the expected risk
- $h_I = argmin_{h \in H} R_I(h)$ be function in H that minimized the empirical risk

Error Decomposition

$$E[R_I(h) - R_I(\hat{h})] = E[R(h^*) - R(\hat{h})] + E[R(h_I) - R(\hat{h})]$$

- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H

Recall Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = argmin_h R(h)$ be function that minimized the expected risk
- $h^\star = argmin_{h \in H} R(h)$ be function in H that minimized the expected risk
- $h_I = argmin_{h \in H} R_I(h)$ be function in H that minimized the empirical risk

Error Decomposition

$$E[R_I(h) - R_I(\hat{h})] = E[R(h^*) - R(\hat{h})] + E[R(h_I) - R(\hat{h})]$$

- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H

Recall Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = argmin_h R(h)$ be function that minimized the expected risk
- $h^\star = argmin_{h \in H} R(h)$ be function in H that minimized the expected risk
- $h_I = argmin_{h \in H} R_I(h)$ be function in H that minimized the empirical risk

Error Decomposition

$$E[R_I(h) - R_I(\hat{h})] = E[R(h^*) - R(\hat{h})] + E[R(h_I) - R(\hat{h})]$$

- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H

Recall Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = argmin_h R(h)$ be function that minimized the expected risk
- $h^\star = argmin_{h \in H} R(h)$ be function in H that minimized the expected risk
- $h_I = argmin_{h \in H} R_I(h)$ be function in H that minimized the empirical risk

Error Decomposition

$$E[R_I(h) - R_I(\hat{h})] = E[R(h^*) - R(\hat{h})] + E[R(h_I) - R(\hat{h})]$$

- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H

Recall Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = argmin_h R(h)$ be function that minimized the expected risk
- $h^\star = argmin_{h \in H} R(h)$ be function in H that minimized the expected risk
- $h_I = argmin_{h \in H} R_I(h)$ be function in H that minimized the empirical risk

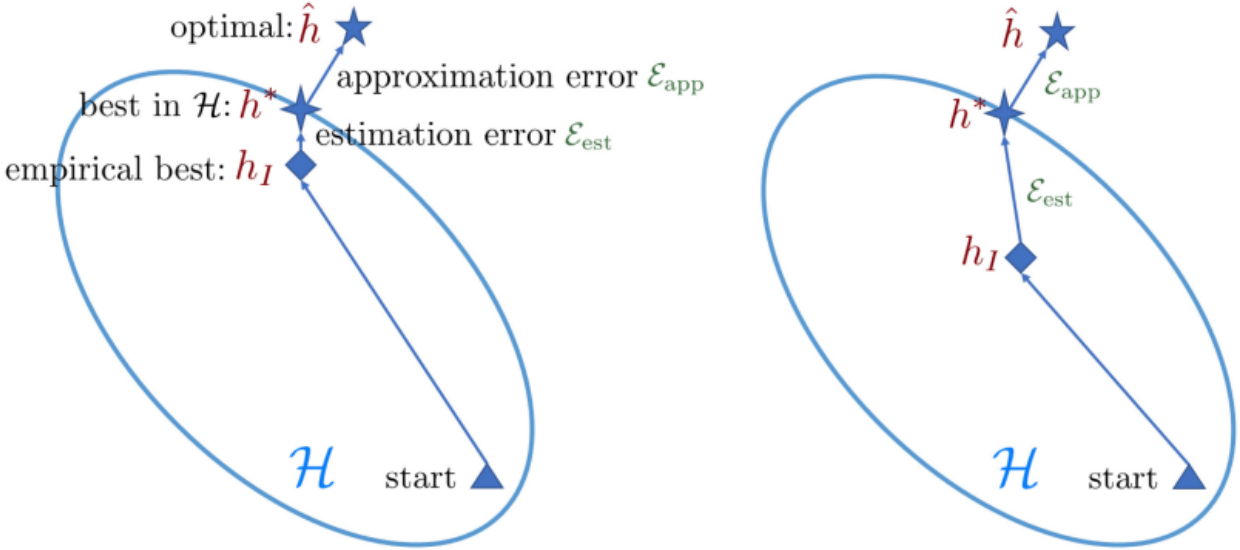
Error Decomposition

$$E[R_I(h) - R_I(\hat{h})] = E[R(h^*) - R(\hat{h})] + E[R(h_I) - R(\hat{h})]$$

- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H

Problem with ZSL/FSL setting

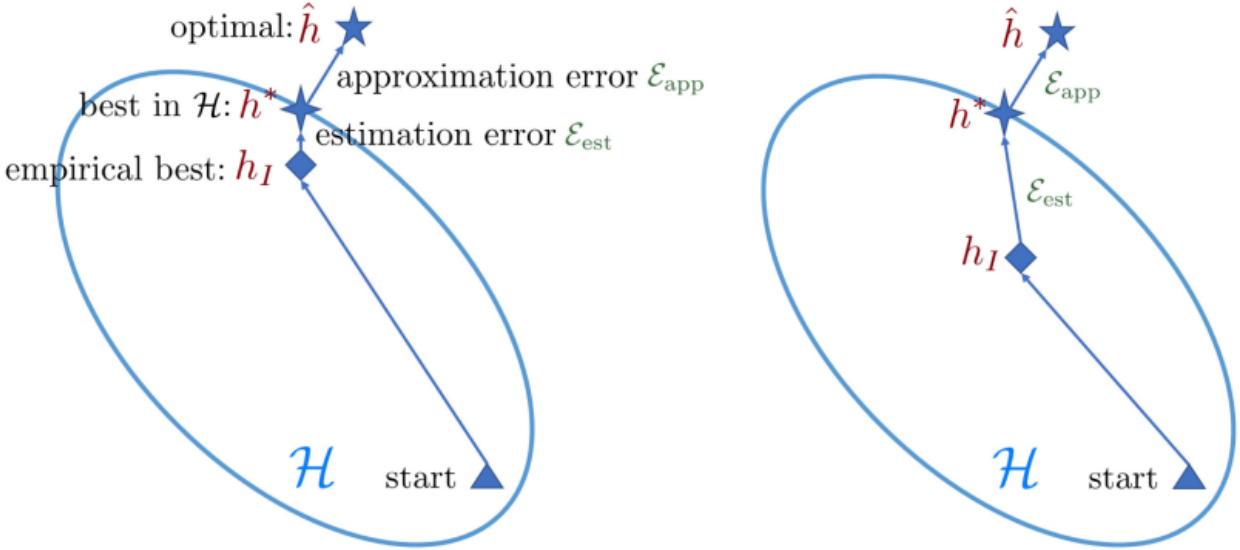
- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H
- FSL, I is very small
- $R_I(h)$ far from being good approximation of expected $R(h)$
- Resultant empirical risk minimizer h_I overfits



(a) Large I (b) Small I

Problem with ZSL/FSL setting

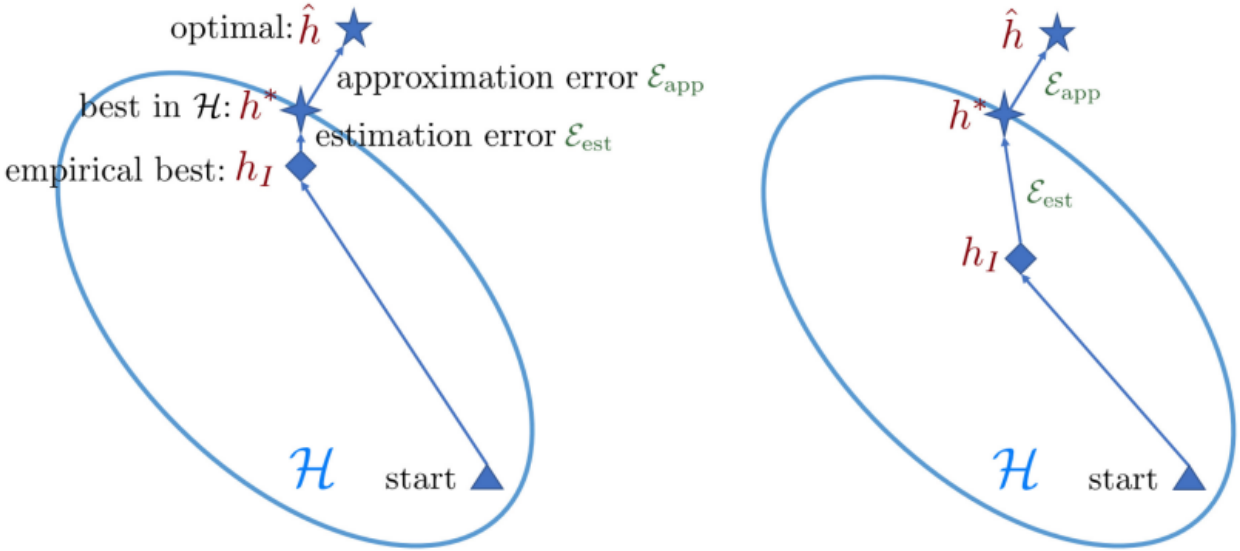
- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H
- FSL, I is very small
- $R_I(h)$ far from being good approximation of expected $R(h)$
- Resultant empirical risk minimizer h_I overfits



(a) Large I (b) Small I

Problem with ZSL/FSL setting

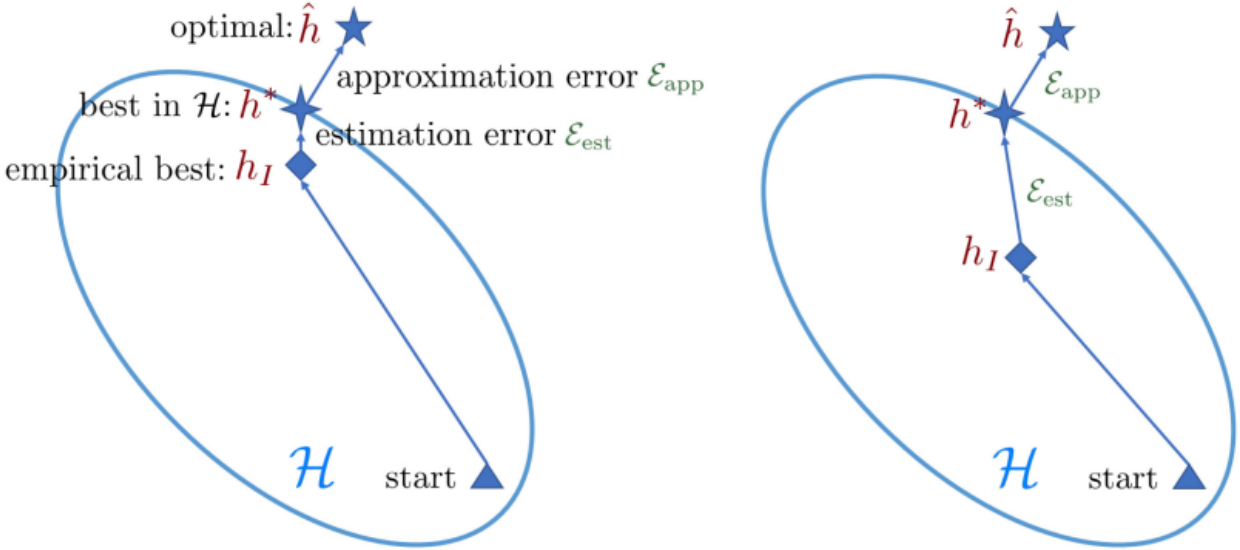
- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H
- FSL, I is very small
- $R_I(h)$ far from being good approximation of expected $R(h)$
- Resultant empirical risk minimizer h_I overfits



(a) Large I (b) Small I

Problem with ZSL/FSL setting

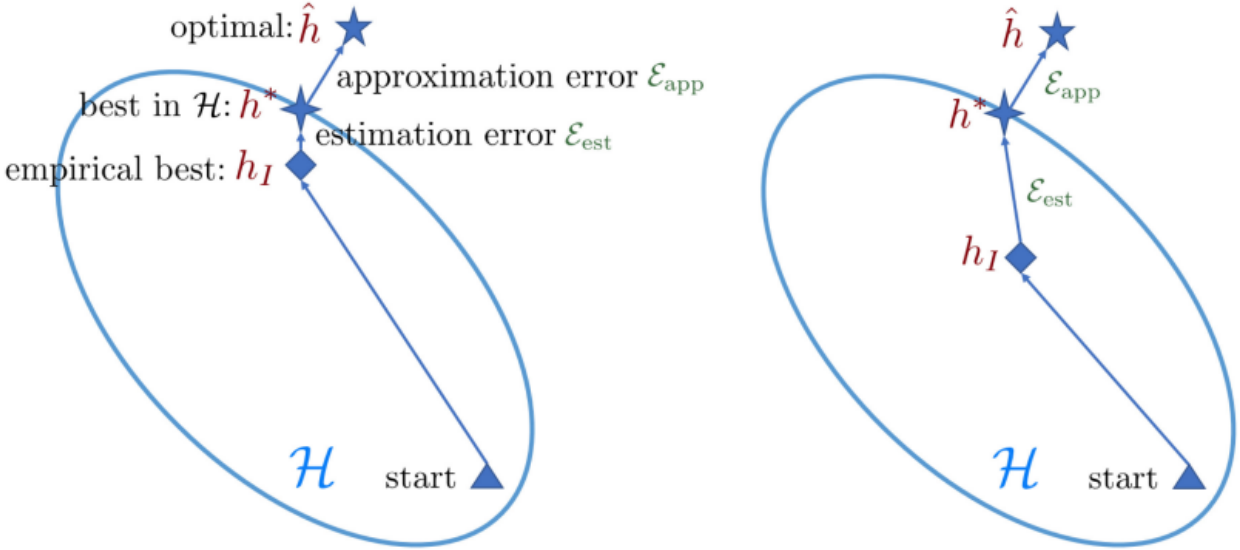
- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H
- FSL, I is very small
- $R_I(h)$ far from being good approximation of expected $R(h)$
- Resultant empirical risk minimizer h_I overfits



(a) Large I (b) Small I

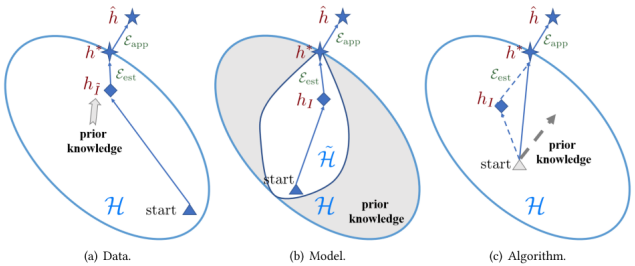
Problem with ZSL/FSL setting

- ϵ_{app} : Approximation error, measure how closely functions in H can approximate the optimal hypothesis $\hat{h}$
- ϵ_{est} : estimation error, measure effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H
- FSL, I is very small
- $R_I(h)$ far from being good approximation of expected $R(h)$
- Resultant empirical risk minimizer h_I overfits



(a) Large I (b) Small I

Addressing ZSL/FSL



(a) Data. (b) Model. (c) Algorithm.

Data

- Learn to augment training data

Model

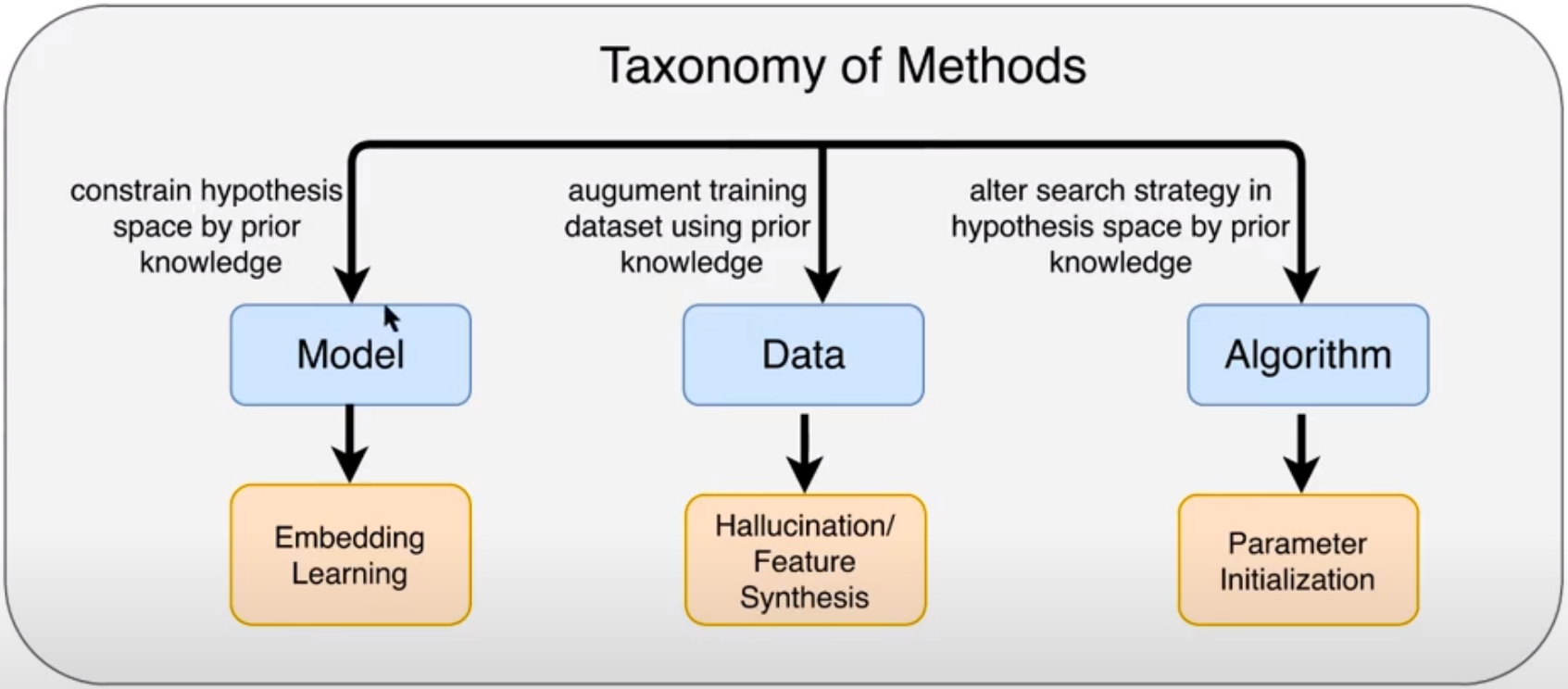
- Constrain complexity of H

Algorithm

- Search for parameters θ for best hypothesis $h^\star$ in H using prior knowledge.

Image credit: WANG et al Generalizing from a Few Examples: A Survey on Few-shot Learning

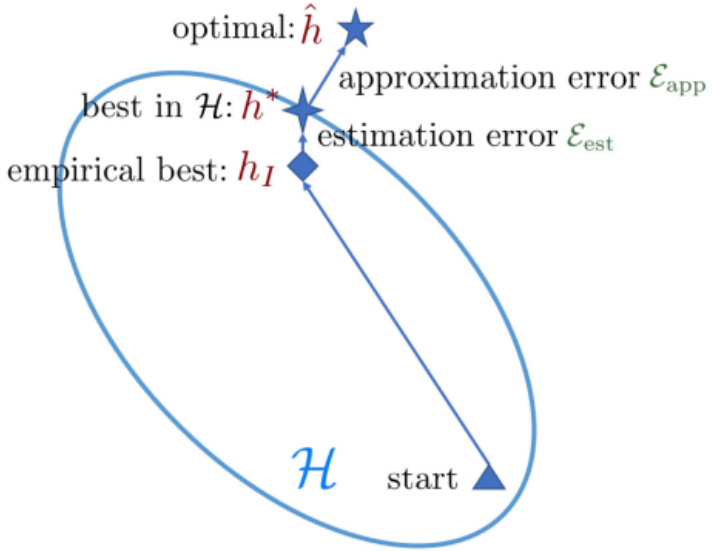
Taxonomy of Methods



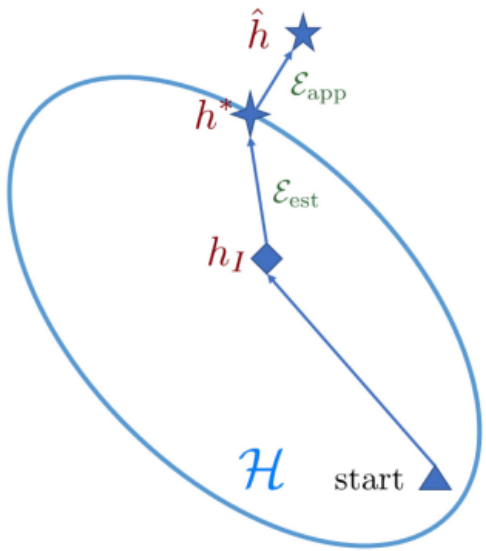
Embedding Learning Methods

Intuition

- Addressing few shot by "learning to compare"
- If model can determine similarity of tow images, it can classify unseen input in relation to labeled instance seen during training



(a) Large I

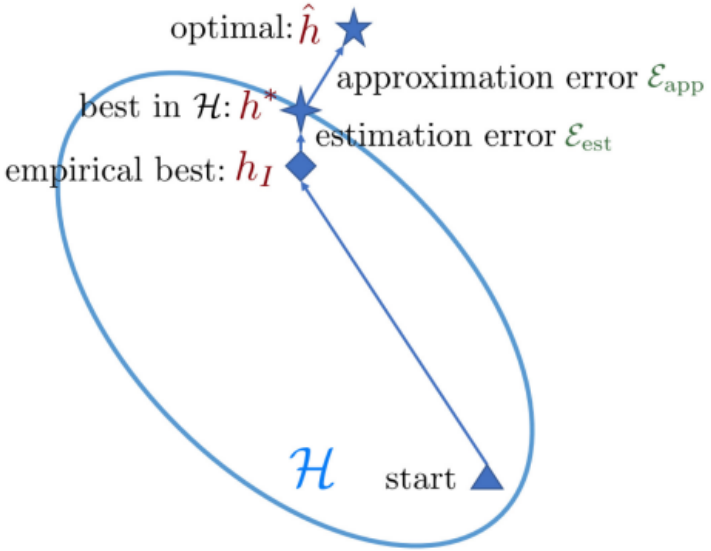


(b) Small I

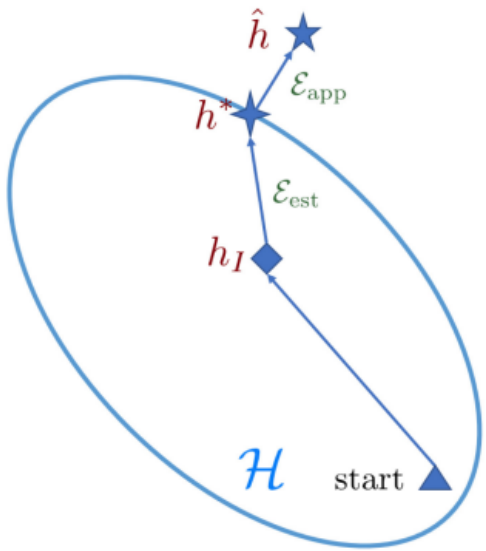
Embedding Learning Methods

Intuition

- Addressing few shot by "learning to compare"
- If model can determine similarity of tow images, it can classify unseen input in relation to labeled instance seen during training



(a) Large I



(b) Small I

Support Set

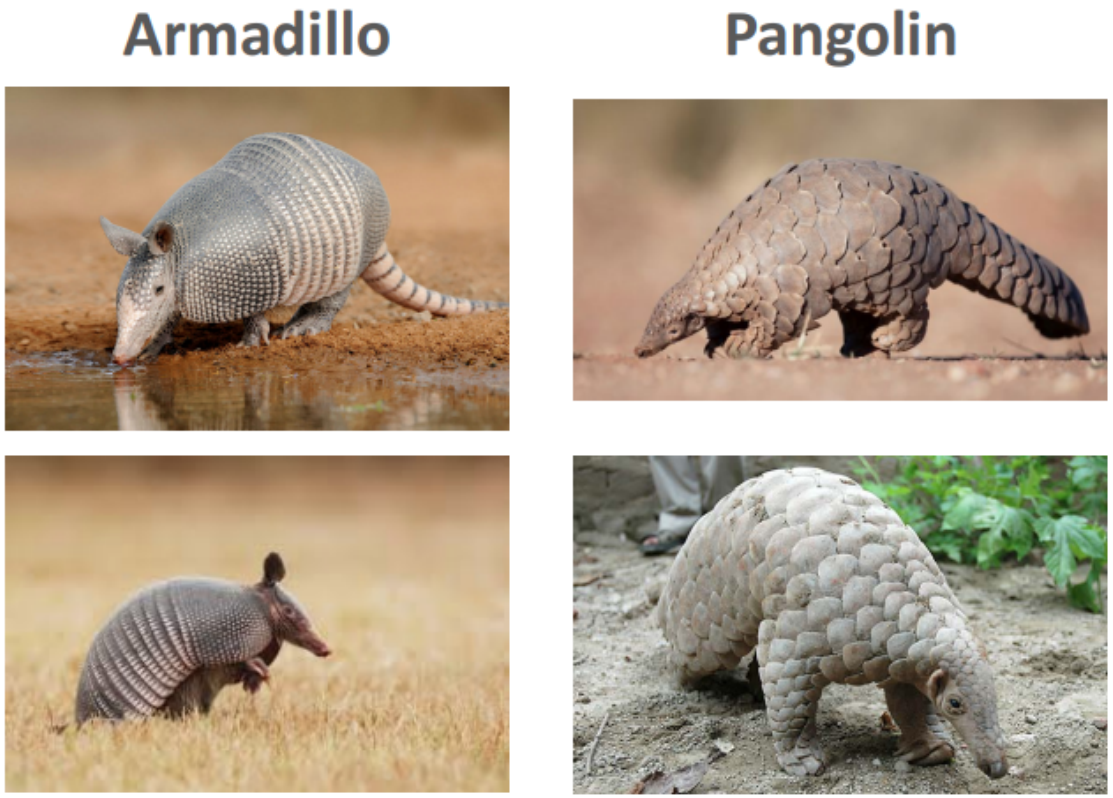


Image credit: Shusen Wang slides

Support Set

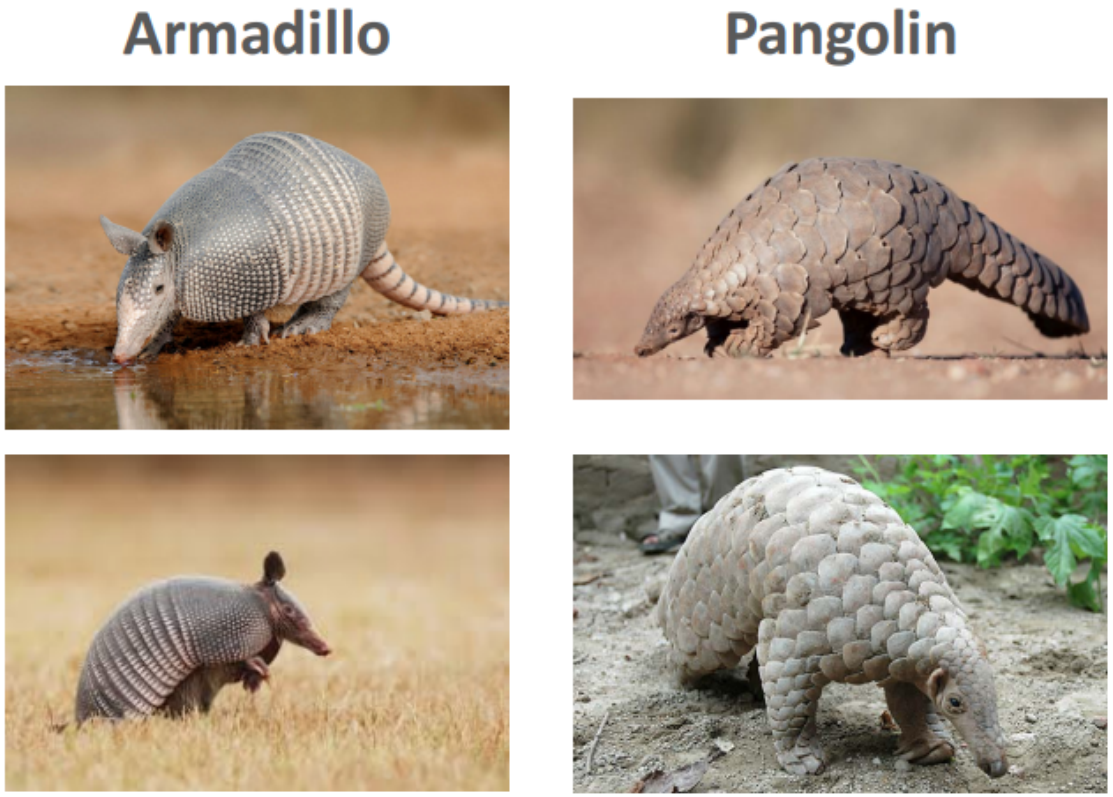


Image credit: Shusen Wang slides

Query

Armadillo or Pangolin?

Training Set

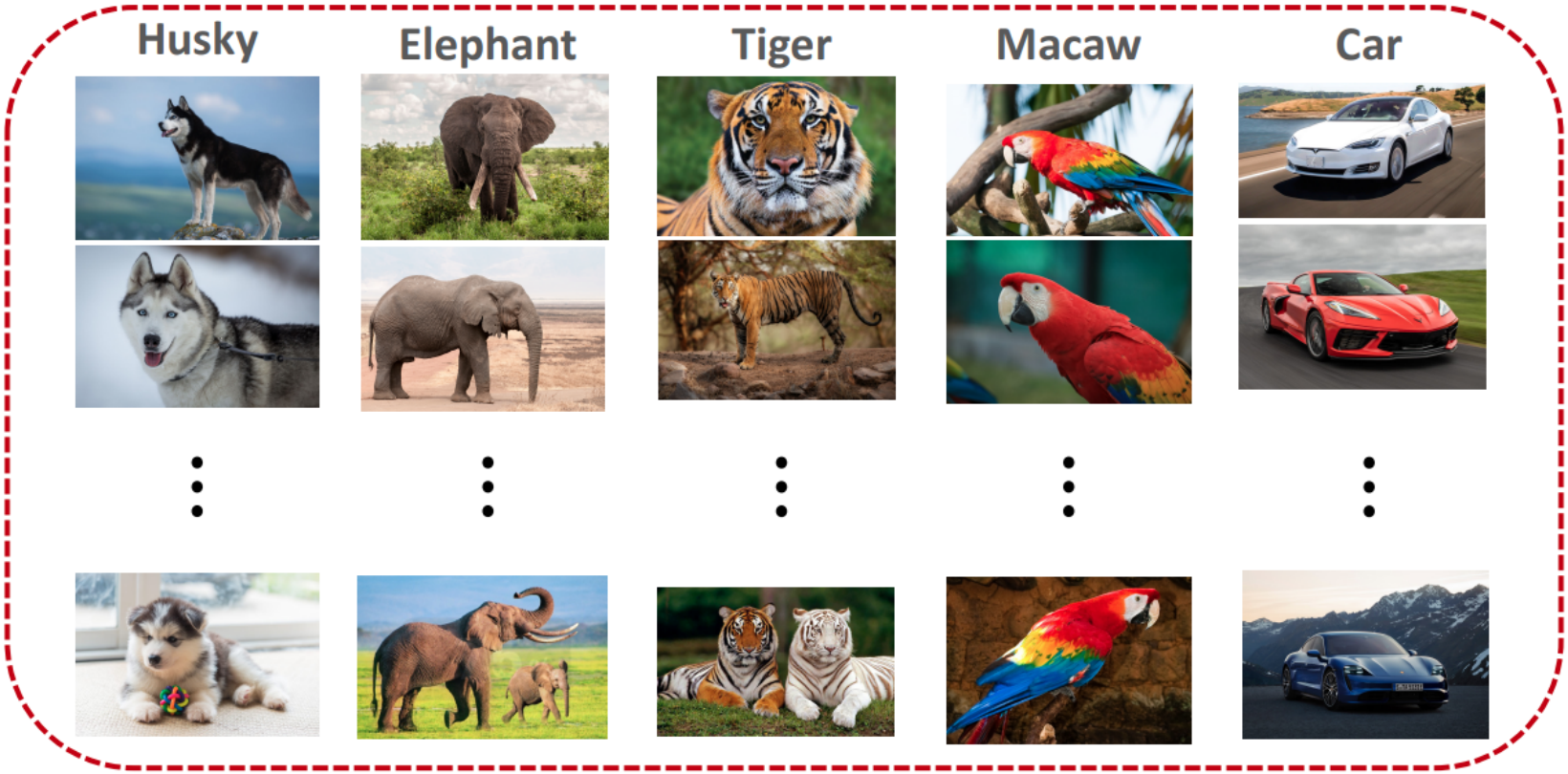


Image credit: Shusen Wang slides

Are they the same kind of animal?

Image credit: Shusen Wang slides

Are they the same kind of animal?

Image credit: Shusen Wang slides

Are they the same kind of animal?

Image credit: Shusen Wang slides

Query:

Image credit: Shusen Wang slides

Meta Learning

- Few-shot learning is a kind of meta learning.
- Meta learning: learn to learn.

Meta Learning

- Few-shot learning is a kind of meta learning.
- Meta learning: learn to learn.

Image credit: Shusen Wang slides

Give him the cards:

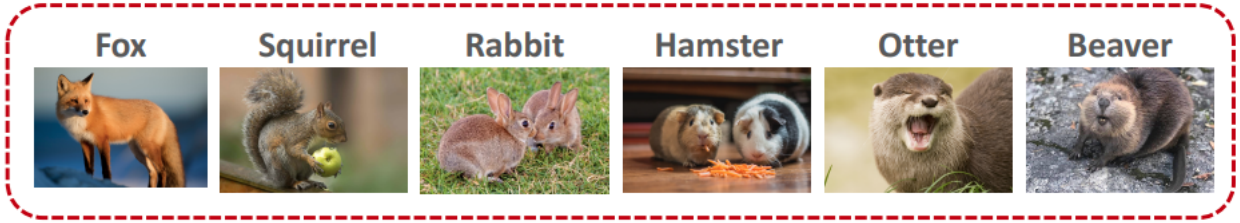
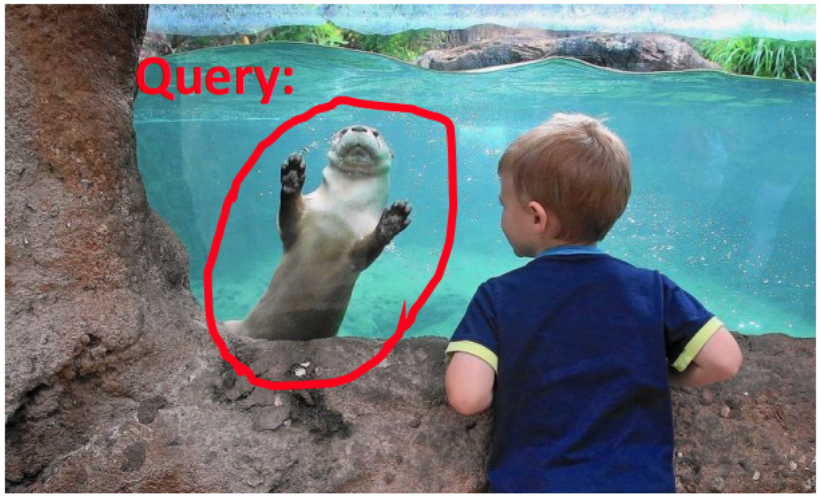


Image credit: Shusen Wang slides



Support set:

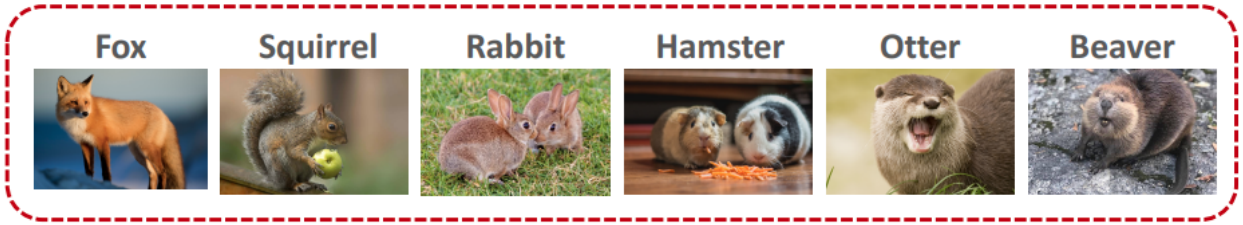
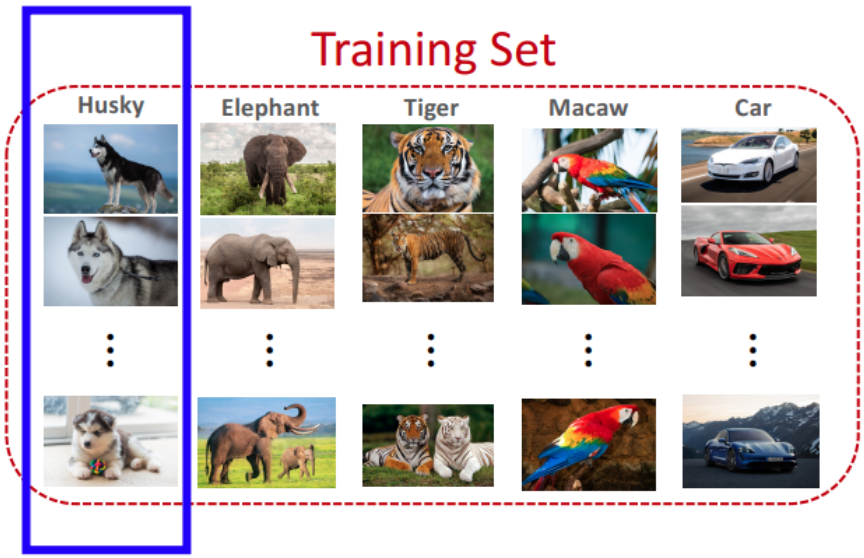


Image credit: Shusen Wang slides

Supervised Learning vs. Few-Shot Learning

Traditional supervised learning

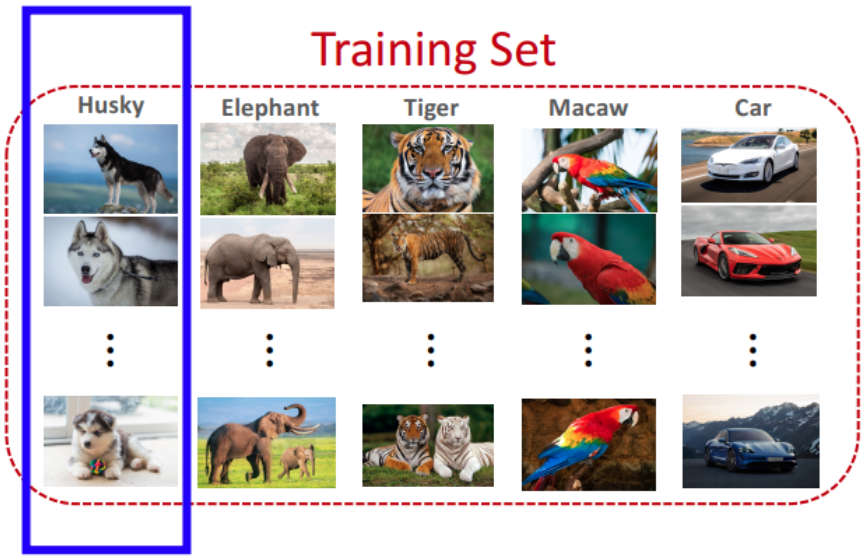
- Test samples are never seen before.
- Test samples are from known classes.



Supervised Learning vs. Few-Shot Learning

Traditional supervised learning

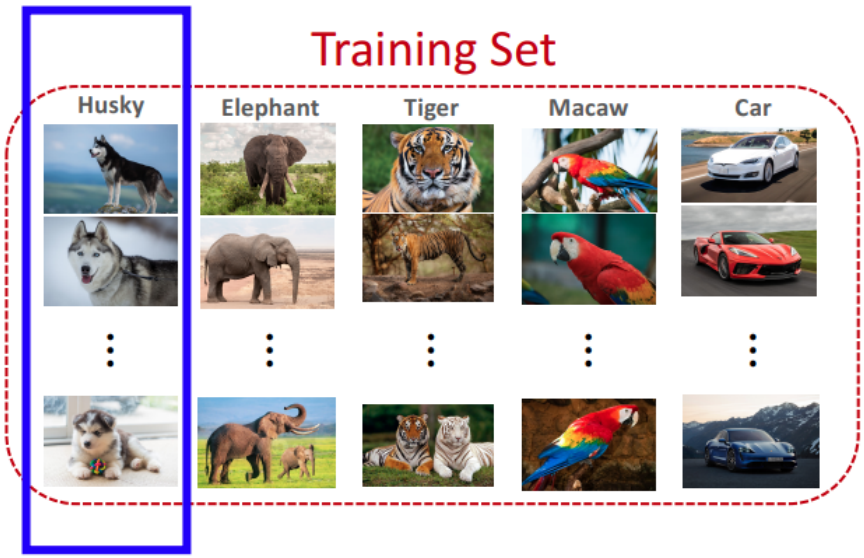
- Test samples are never seen before.
- Test samples are from known classes.



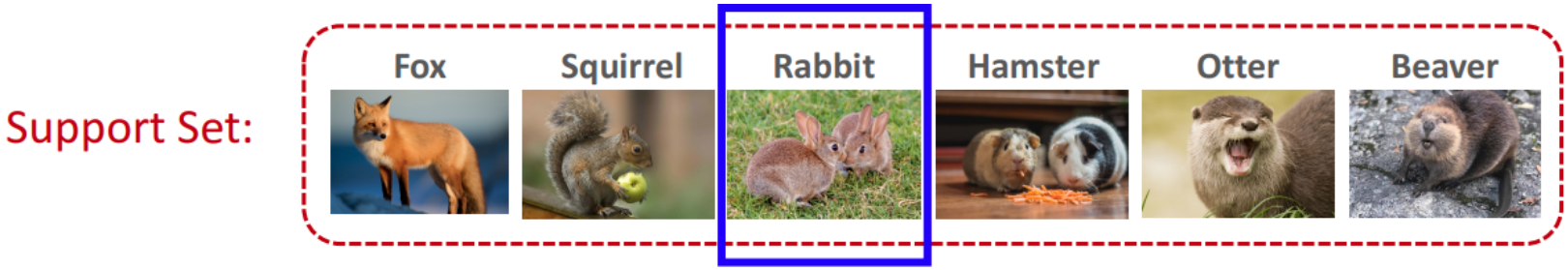
Supervised Learning vs. Few-Shot Learning

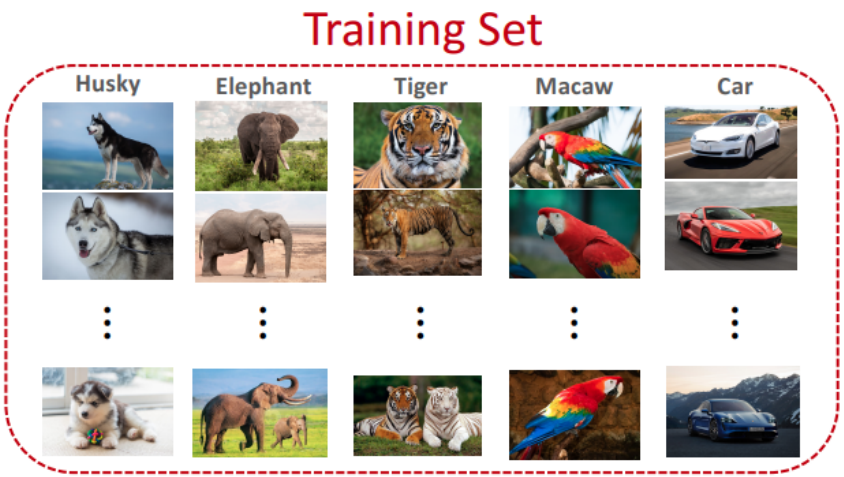
Few-shot learning

- Query samples are never seen before.
- Query samples are from unknown classes.

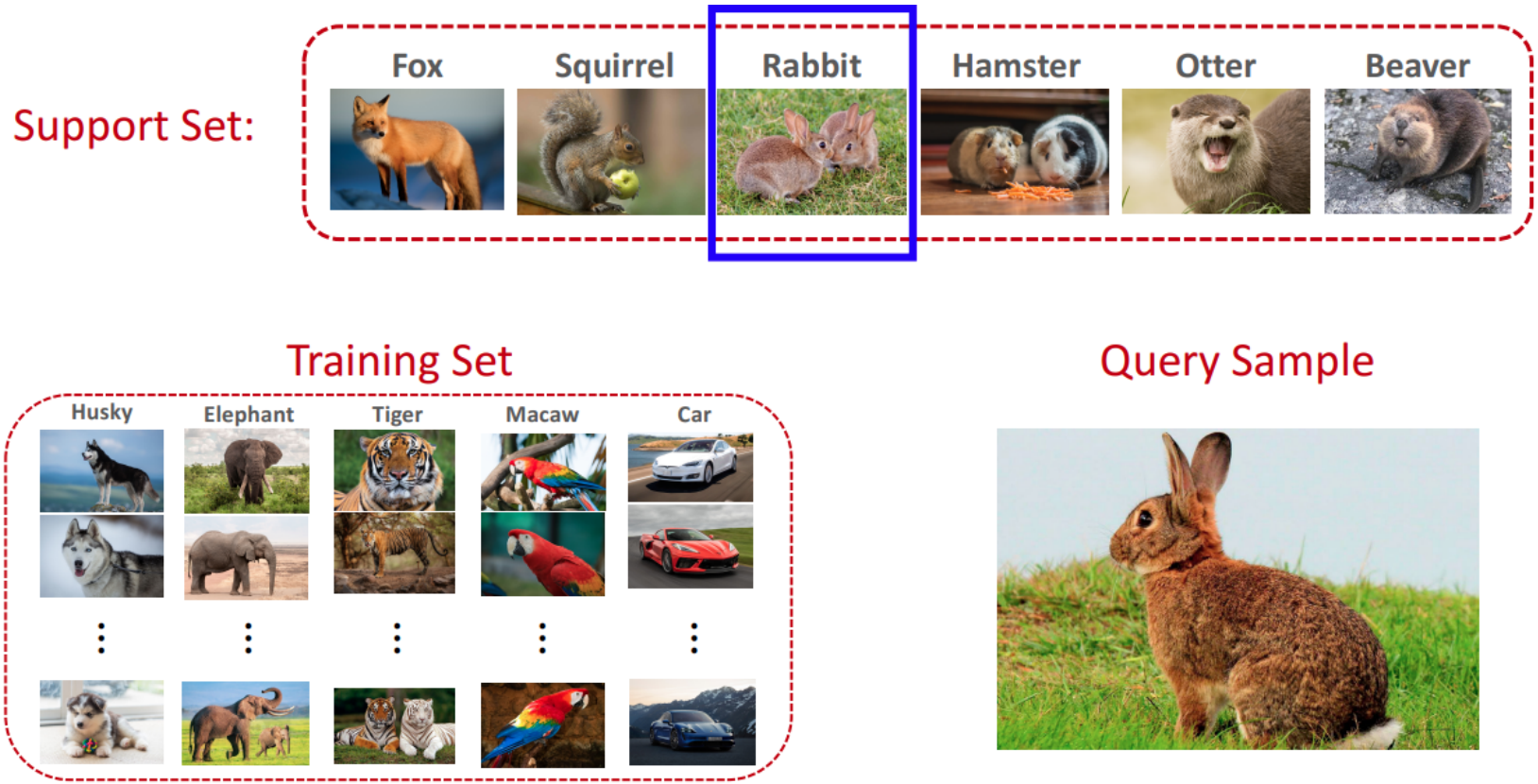


Training Set, Support Set, and Query



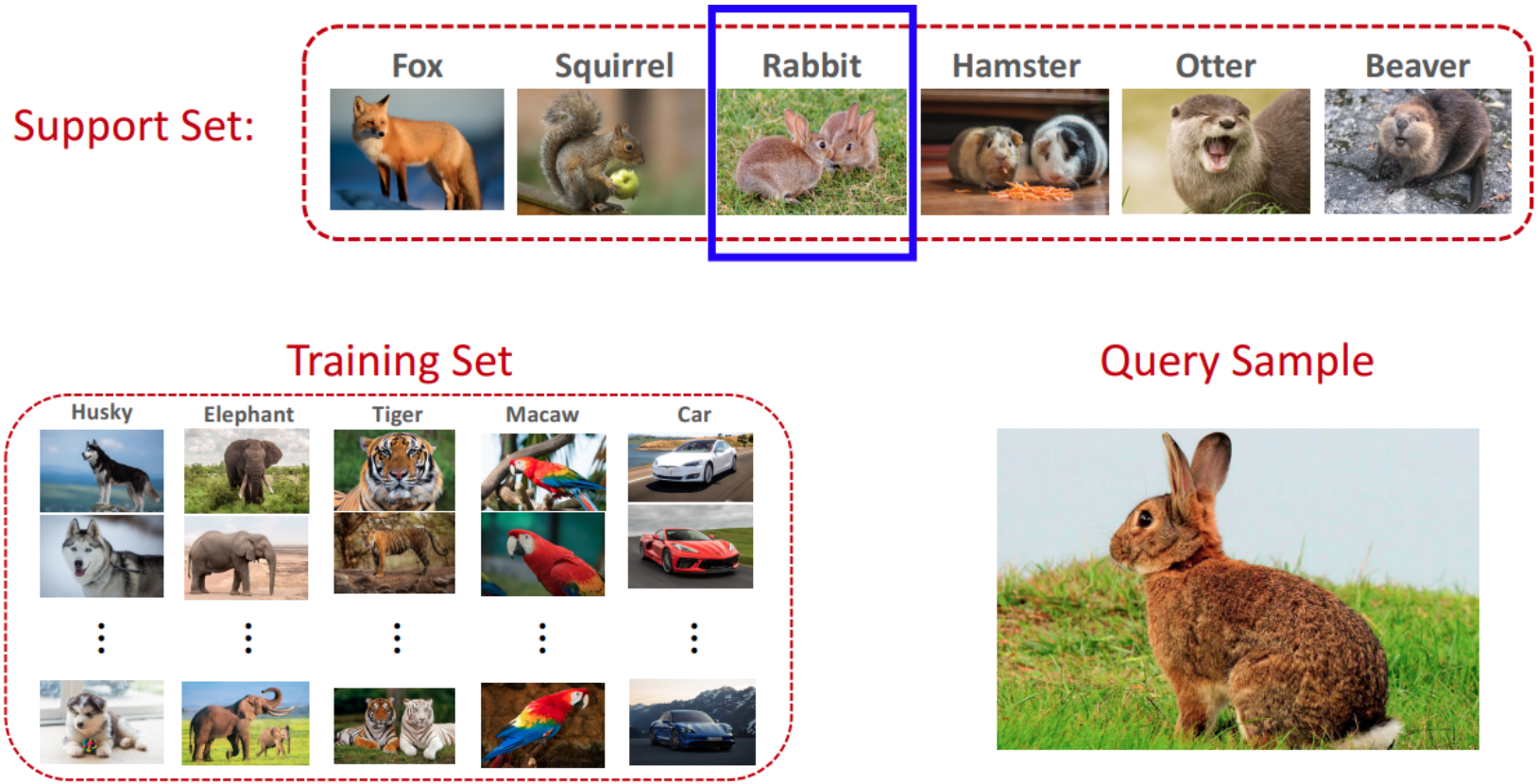


k-way n-shot Support Set



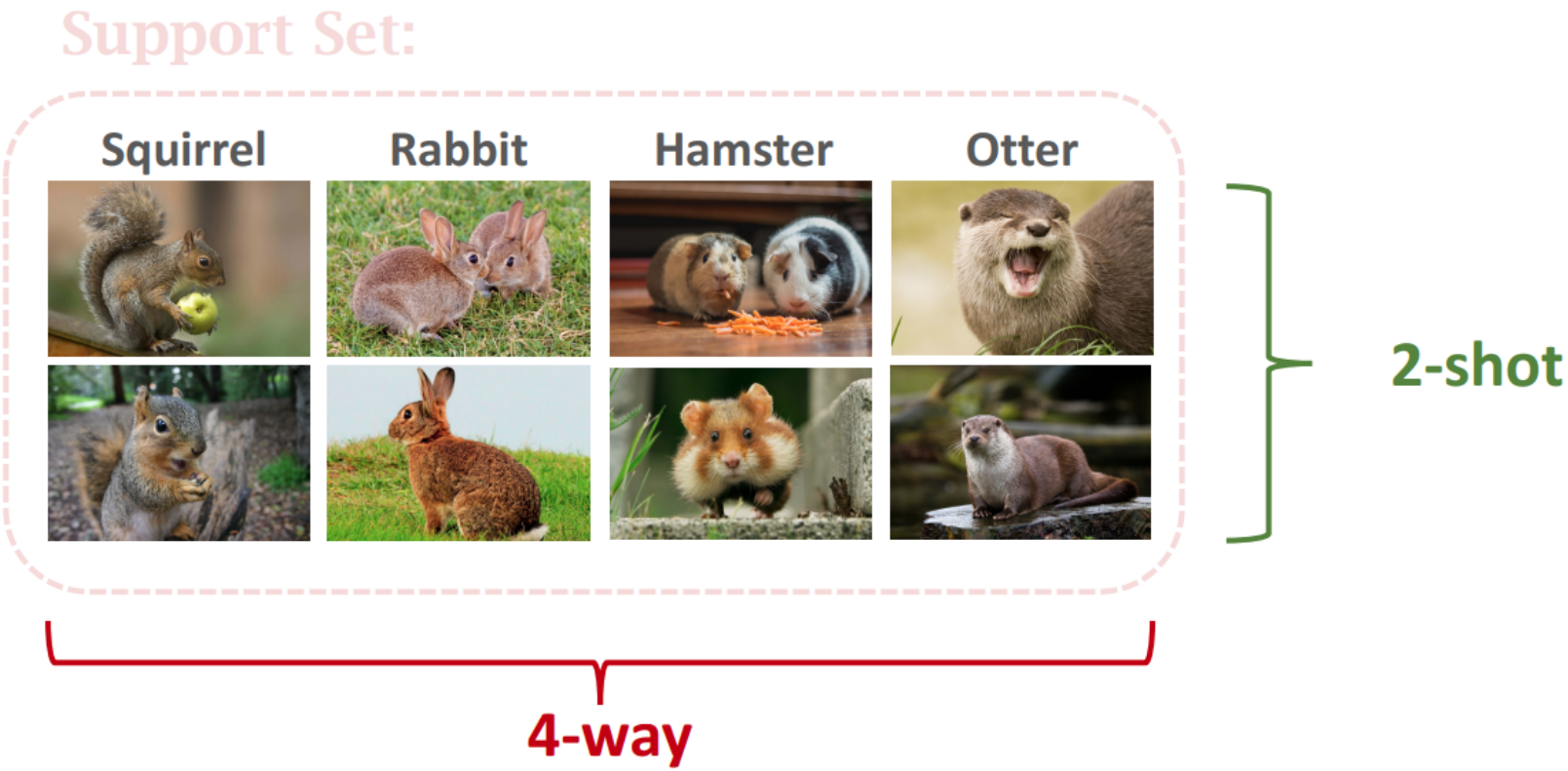
- k-way: the support set has K classes.
- n-shot: every class has n samples.

k-way n-shot Support Set



- k-way: the support set has K classes.
- n-shot: every class has n samples.

k-way n-shot Support Set



Prediction Accuracy

- Acc v/s Number of ways
- Acc v/s Number of shots
- Which is easier 2-ways or 3-ways ?
- Which is easier 2-shot or 3-shots ?

Prediction Accuracy

- Acc v/s Number of ways
- Acc v/s Number of shots
- Which is easier 2-ways or 3-ways ?
- Which is easier 2-shot or 3-shots ?

Prediction Accuracy

- Acc v/s Number of ways
- Acc v/s Number of shots
- Which is easier 2-ways or 3-ways ?
- Which is easier 2-shot or 3-shots ?

Prediction Accuracy

- Acc v/s Number of ways
- Acc v/s Number of shots
- Which is easier 2-ways or 3-ways ?
- Which is easier 2-shot or 3-shots ?

Idea: Learn a Similarity Function

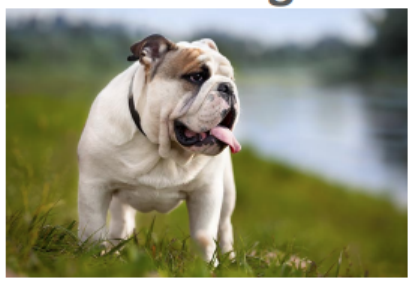
Basic idea

- Learn a similarity function: $sim(x, x')$
- Ideally $sim(x_1, x_2) = 1$, $sim(x_1, x_3) = 0$, $sim(x_2, x_3) = 0$

Bulldog

$\mathbf{x}_1$

Bulldog



$\mathbf{x}_2$

Fox

$\mathbf{x}_3$

Idea: Learn a Similarity Function

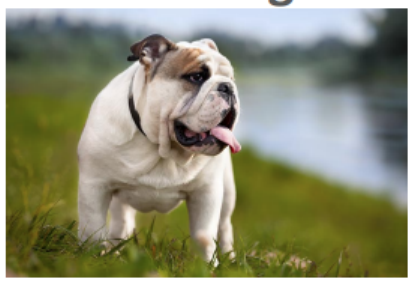
Basic idea

- Learn a similarity function: $sim(x, x')$
- Ideally $sim(x_1, x_2) = 1$, $sim(x_1, x_3) = 0$, $sim(x_2, x_3) = 0$

Bulldog

$\mathbf{x}_1$

Bulldog



$\mathbf{x}_2$

Fox

$\mathbf{x}_3$

Basic idea

- First, learn a similarity function from large-scale training dataset.

Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

Support Set:

Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

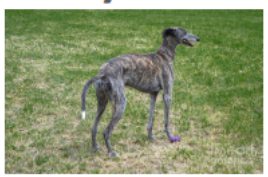
Support Set:

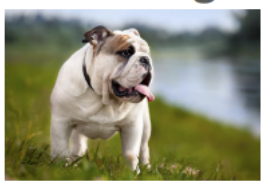
Basic idea

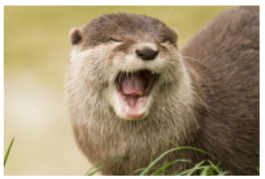
What is in the image?

Query:

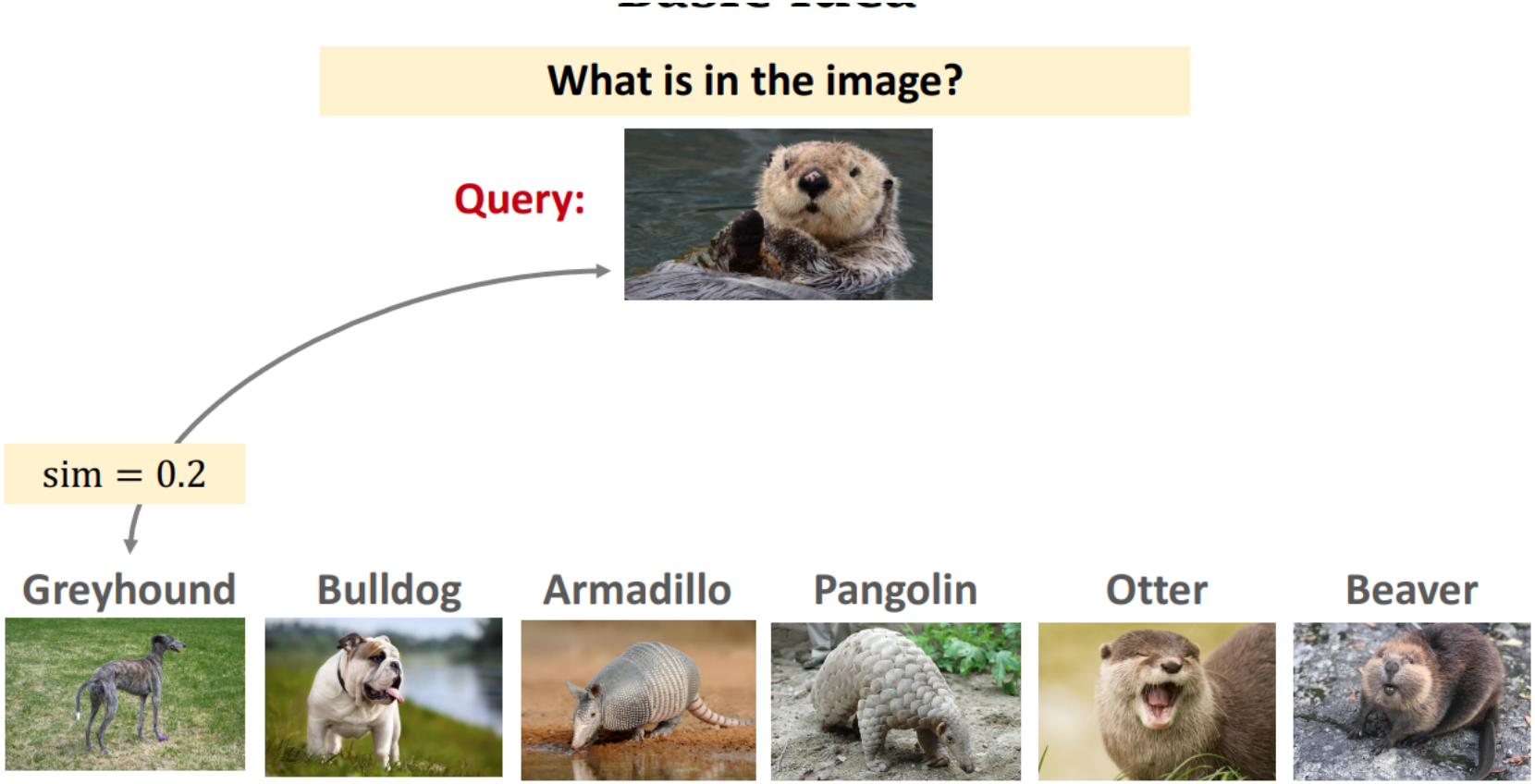
Support Set:



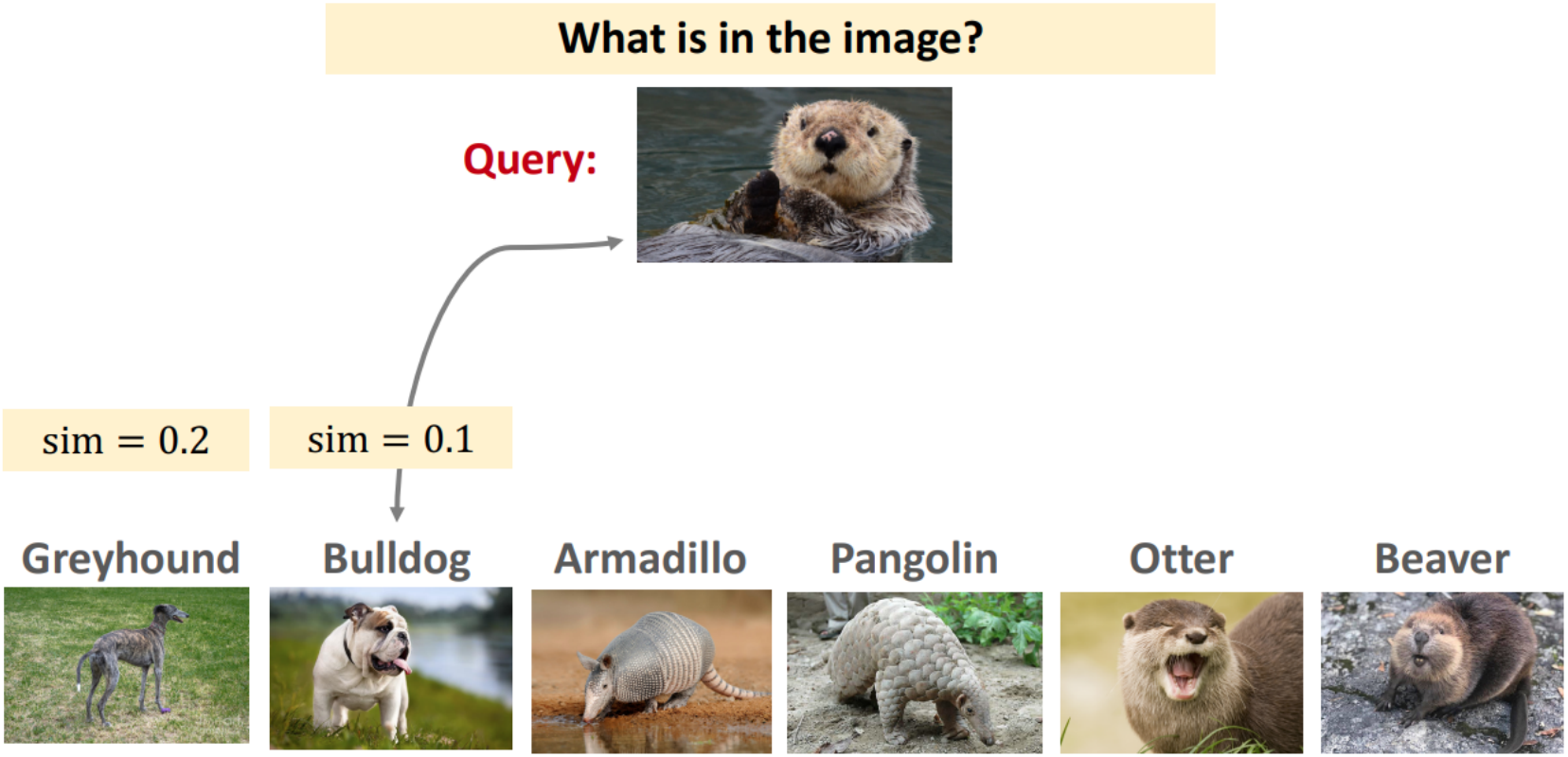




Basic idea


What is in the image?
Query:
sim = 0.2
Greyhound
Bulldog
Armadillo
Pangolin
Otter
Beaver

Basic idea


What is in the image?
Query:
sim = 0.2
sim = 0.1
Greyhound
Bulldog
Armadillo
Pangolin
Otter
Beaver

Basic idea

What is in the image?

Query:

sim = 0.2	sim = 0.1	sim = 0.03	sim = 0.05	sim = 0.7	sim = 0.5
Greyhound	Bulldog	Armadillo	Pangolin	Otter	Beaver



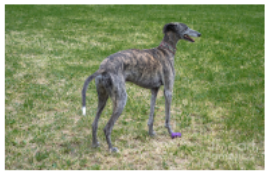



Basic idea

What is in the image?

Query:

sim = 0.2	sim = 0.1	sim = 0.03	sim = 0.05	sim = 0.7	sim = 0.5
Greyhound	Bulldog	Armadillo	Pangolin	Otter	Beaver






Training Data

Positive Samples

( ,  , 1)

( ,  , 1)

( ,  , 1)

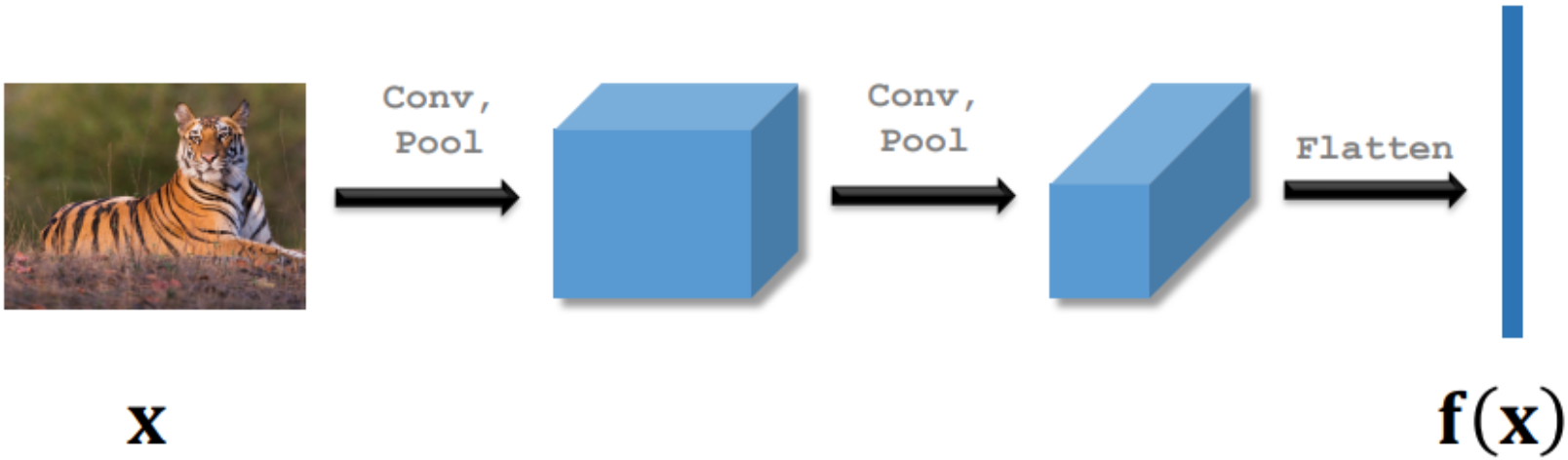
Negative Samples

( ,  , 0)

( ,  , 0)

( ,  , 0)

CNN for Feature Extraction



Training Siamese Network

$\mathbf{x}_1$

$\mathbf{x}_2$

Training Siamese Network

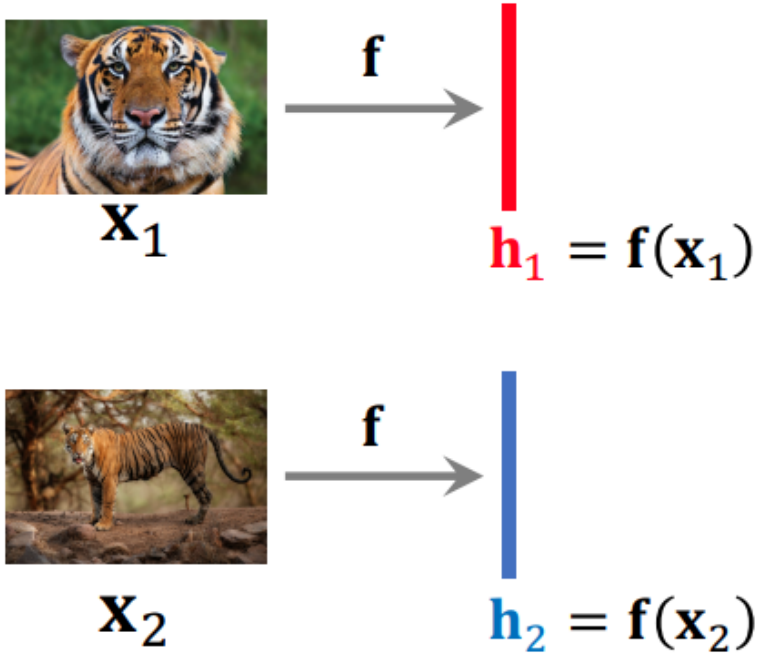
$\mathbf{f}$

$\mathbf{x}_1$

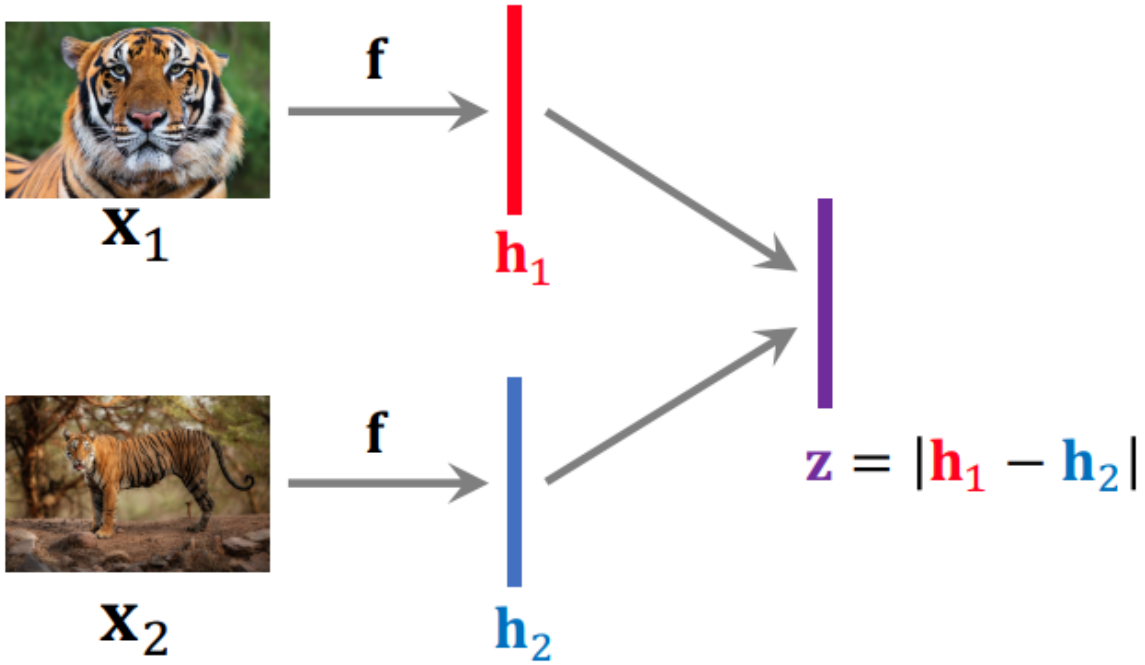
$\mathbf{f}$

$\mathbf{x}_2$

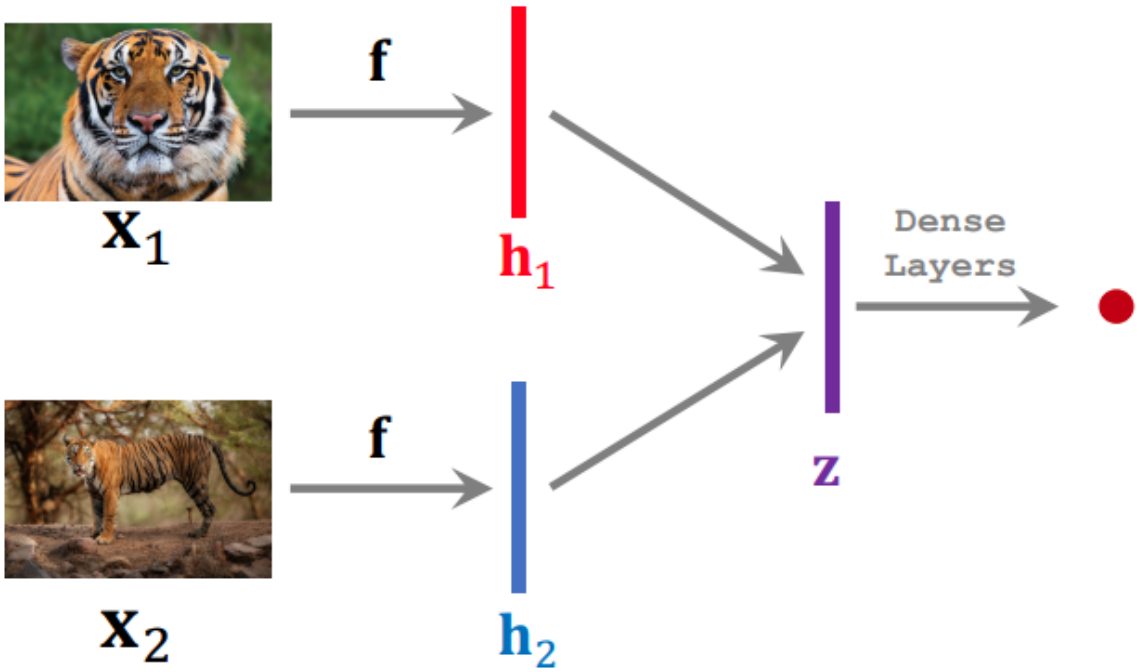
Training Siamese Network



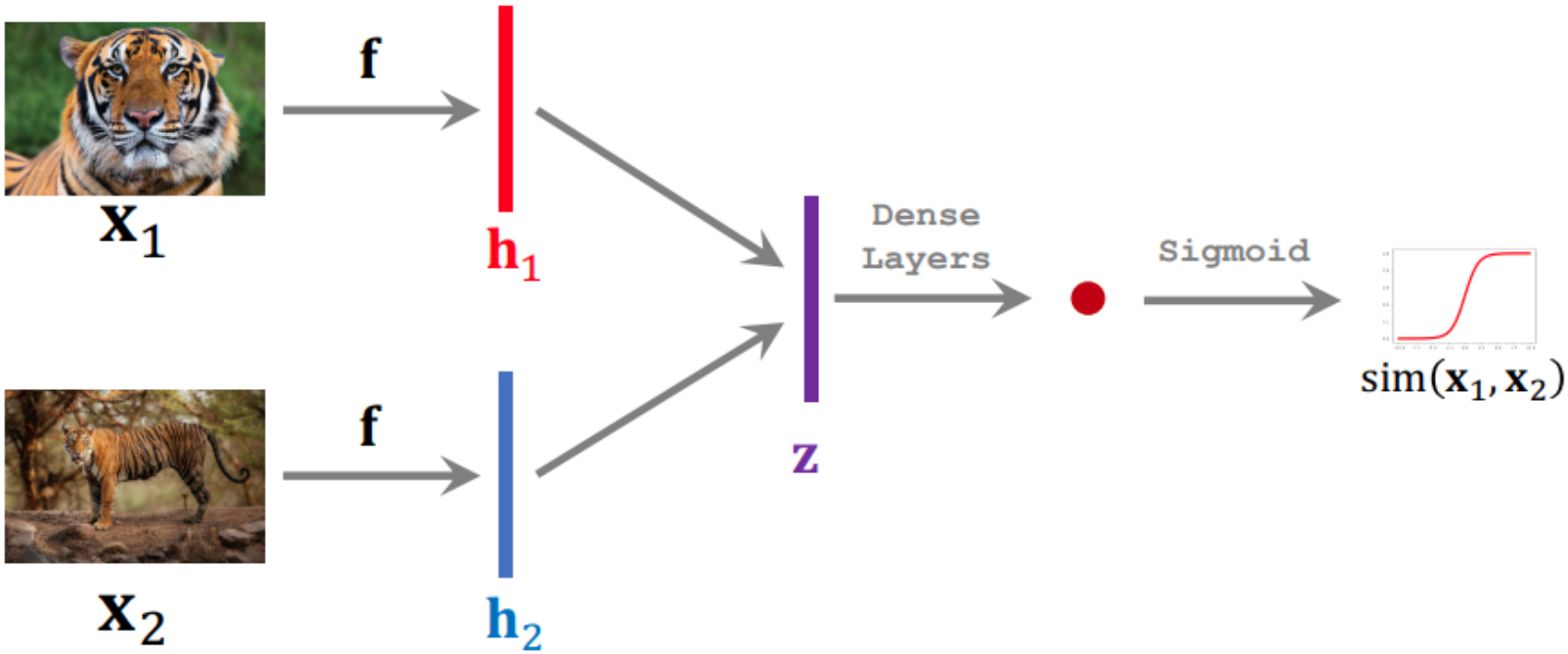
Training Siamese Network



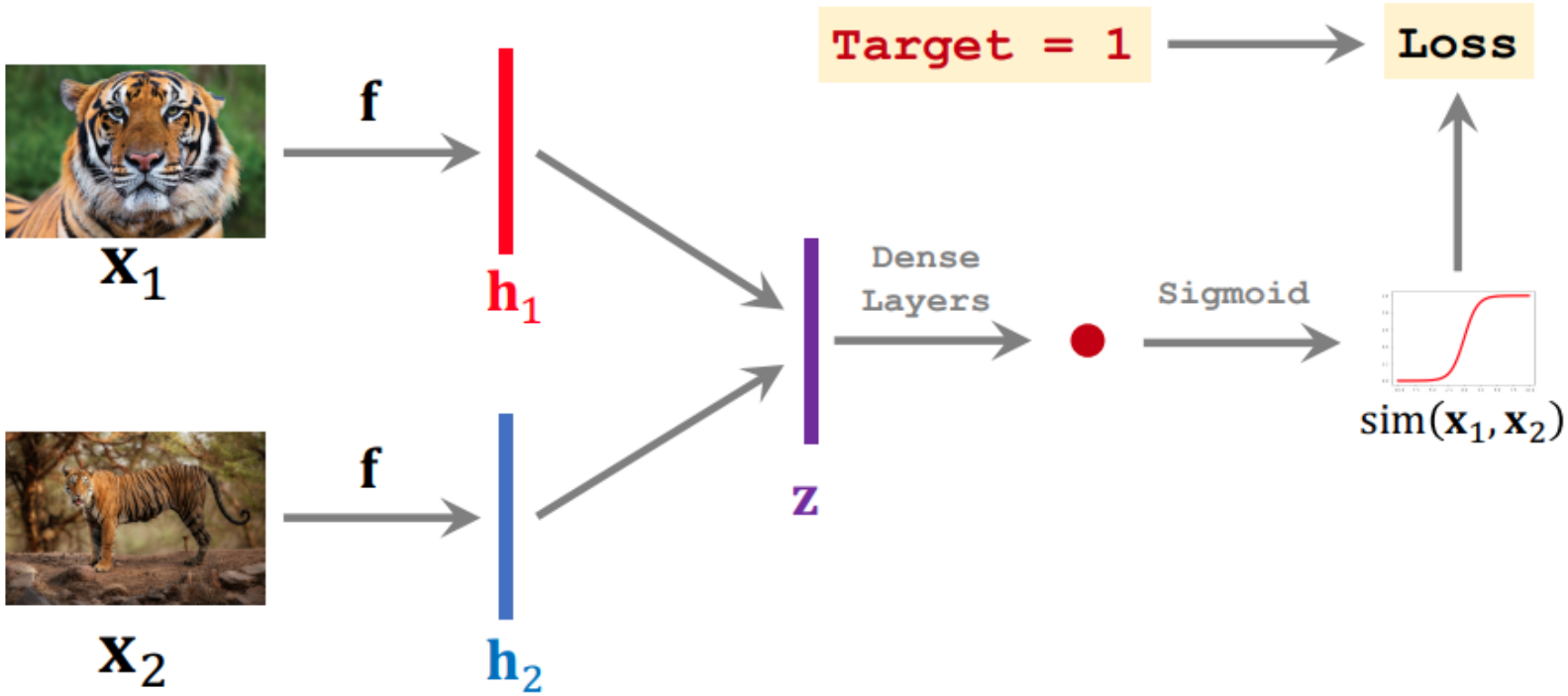
Training Siamese Network



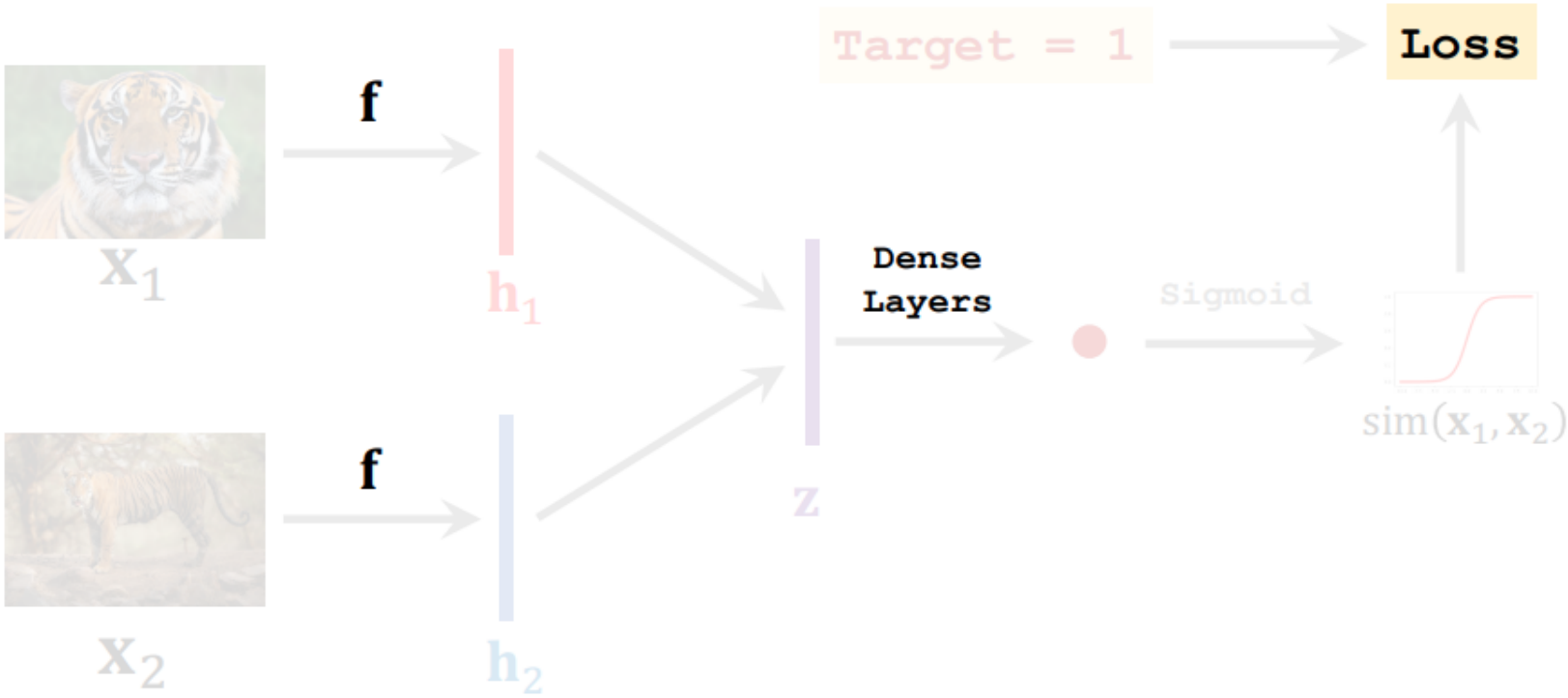
Training Siamese Network



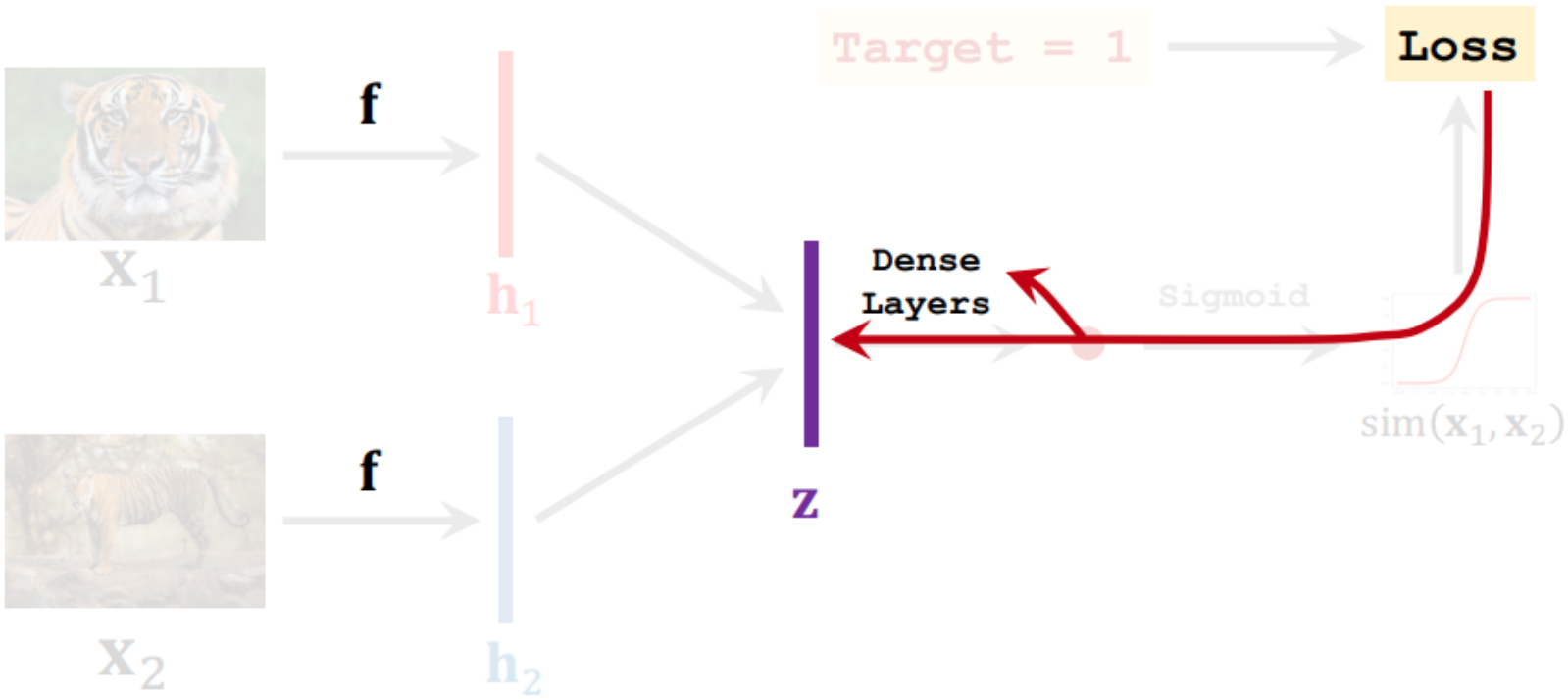
Training Siamese Network



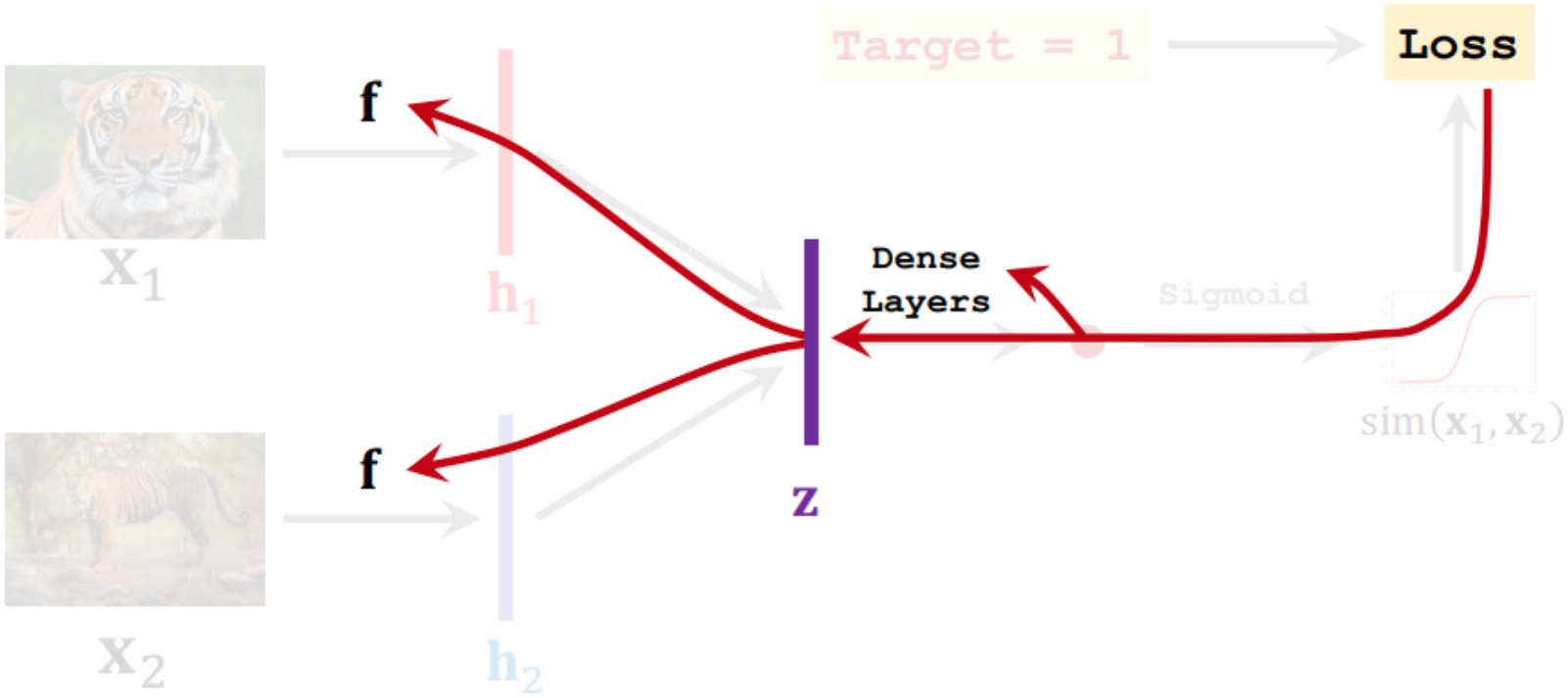
Training Siamese Network



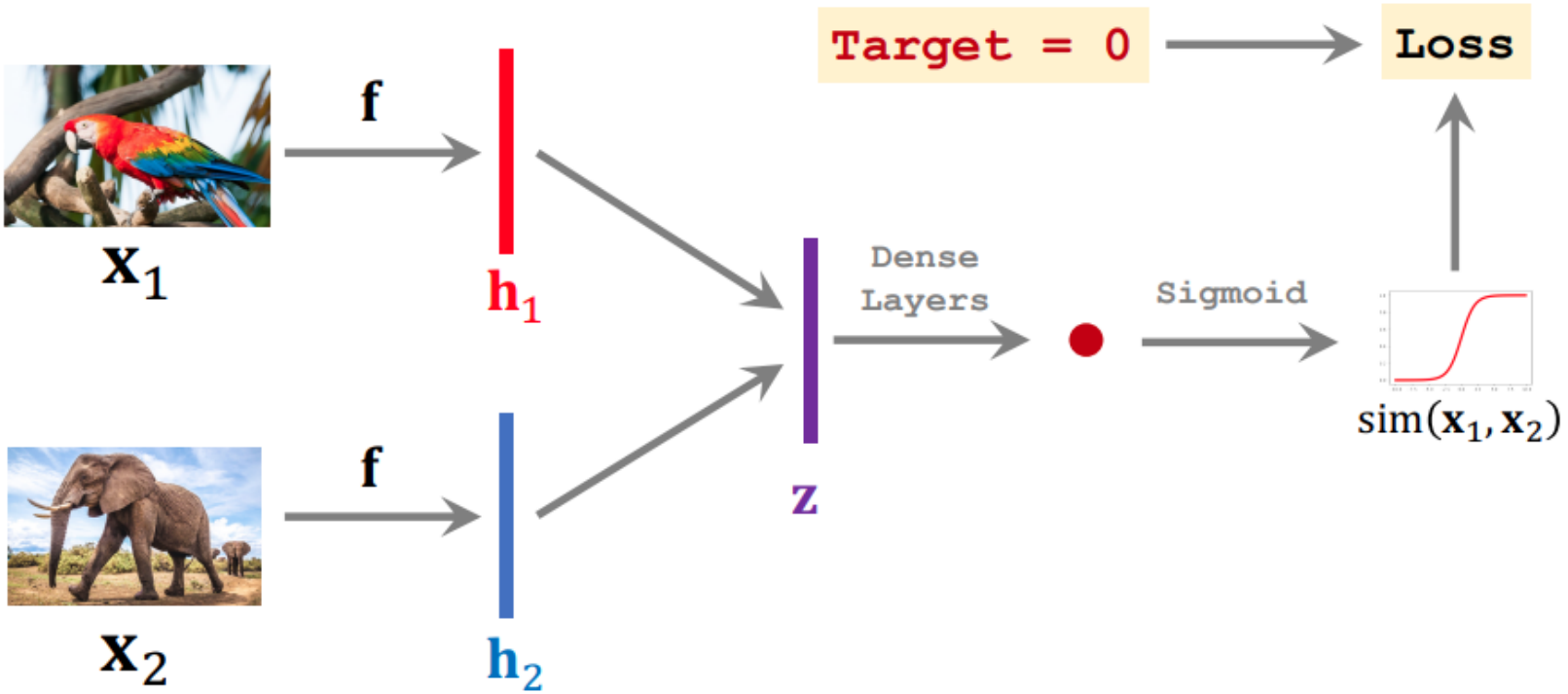
Training Siamese Network



Training Siamese Network

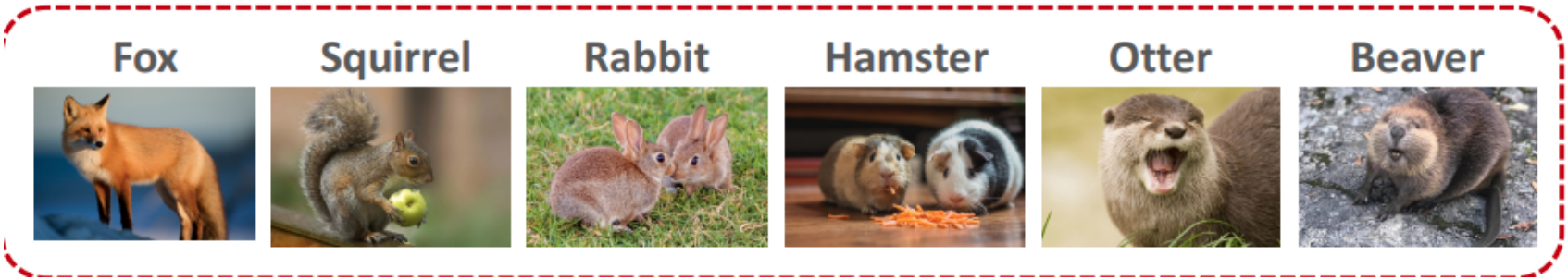


Training Siamese Network



One-Shot Prediction

Support Set:



One-Shot Prediction

One-Shot Prediction

Query:

Support Set:

Fox

Squirrel

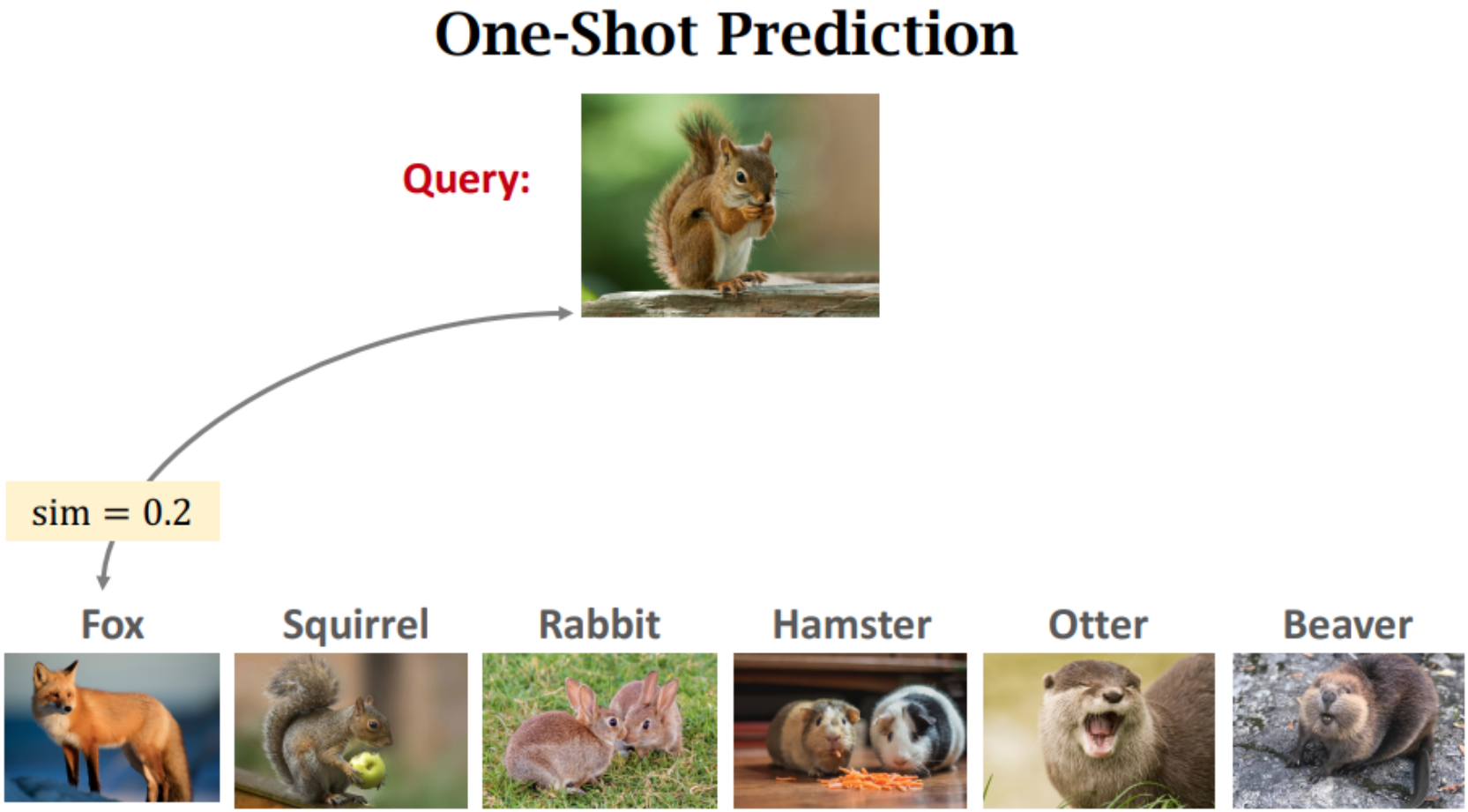
Rabbit

Hamster

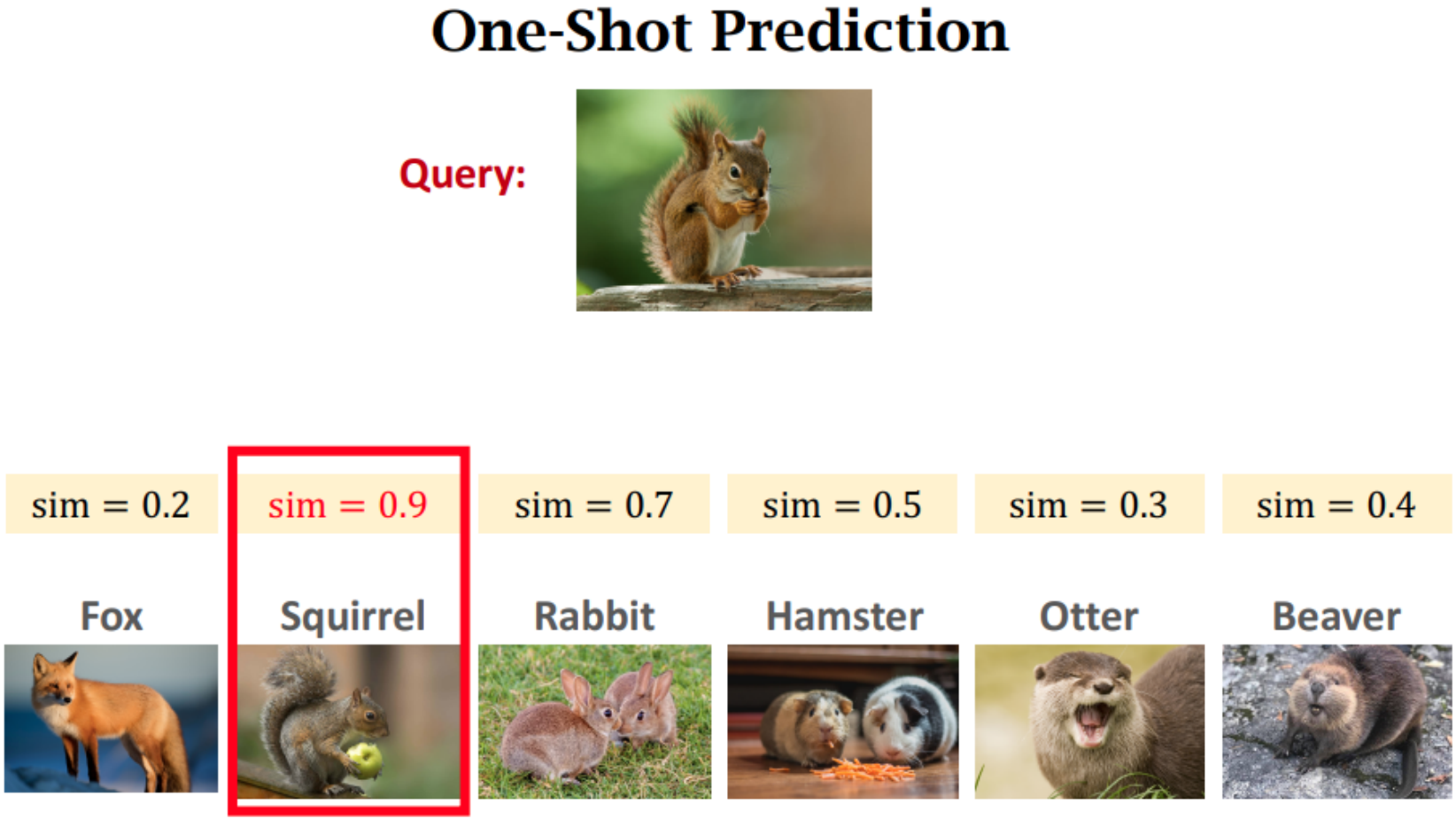
Otter

Beaver

One-Shot Prediction



One-Shot Prediction



Triplet Loss

Data for Training Siamese Network

Data for Training Siamese Network

Data for Training Siamese Network

Data for Training Siamese Network

Data for Training Siamese Network

Data for Training Siamese Network

Triplet Loss

$\mathbf{x}^+$
(positive)

$\mathbf{x}^a$
(anchor)

$\mathbf{x}^-$
(negative)

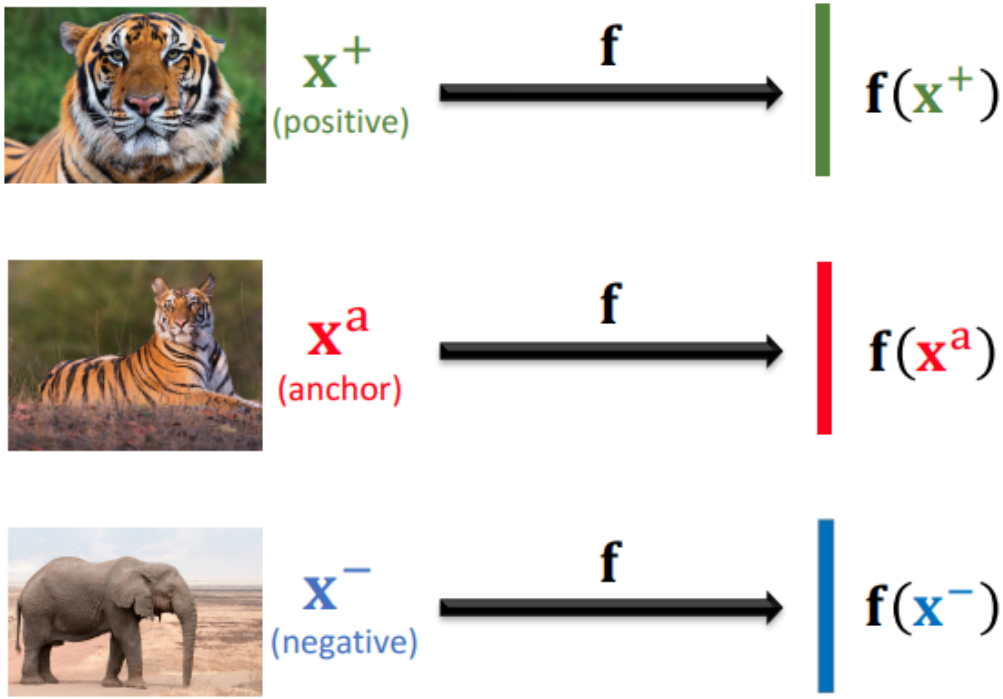
Triplet Loss

$\mathbf{x}^+$ (positive) $\xrightarrow{\mathbf{f}}$

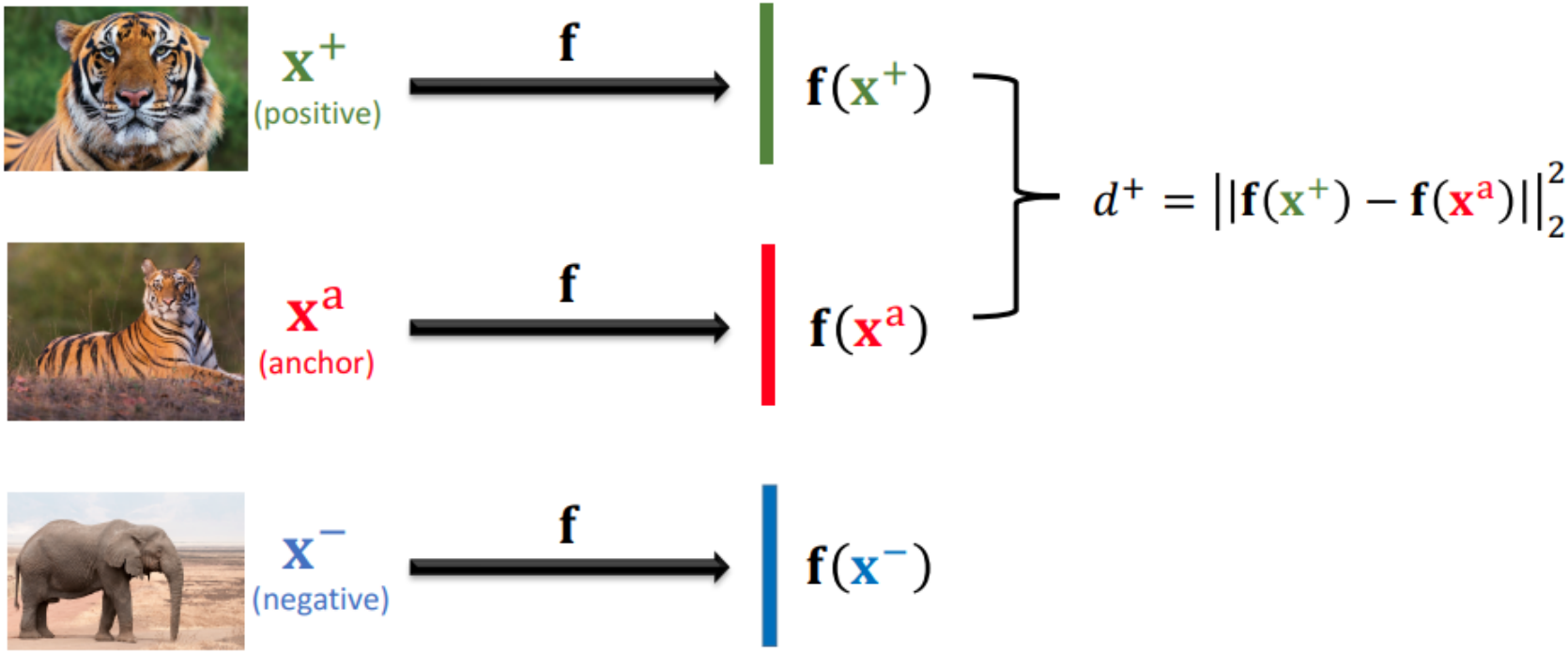
$\mathbf{x}^a$ (anchor) $\xrightarrow{\mathbf{f}}$

$\mathbf{x}^-$ (negative) $\xrightarrow{\mathbf{f}}$

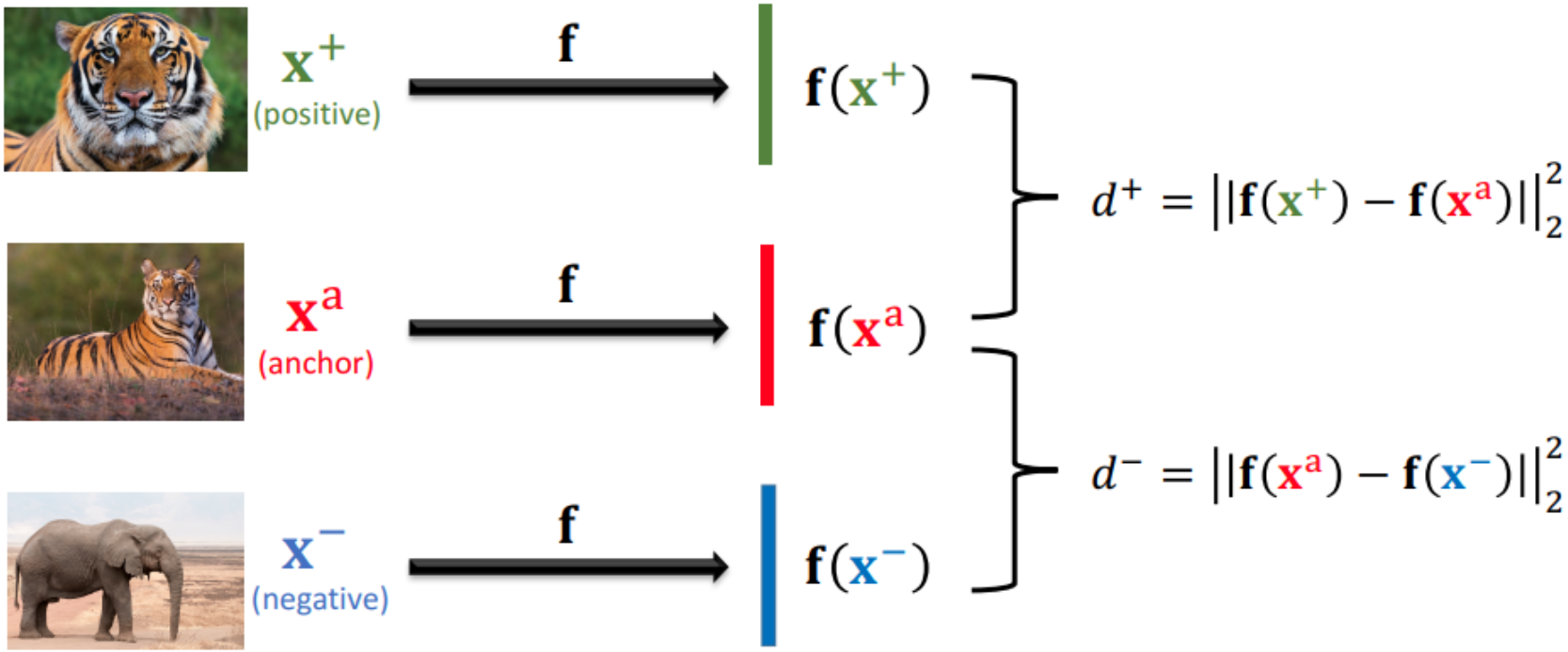
Triplet Loss



Triplet Loss



Triplet Loss

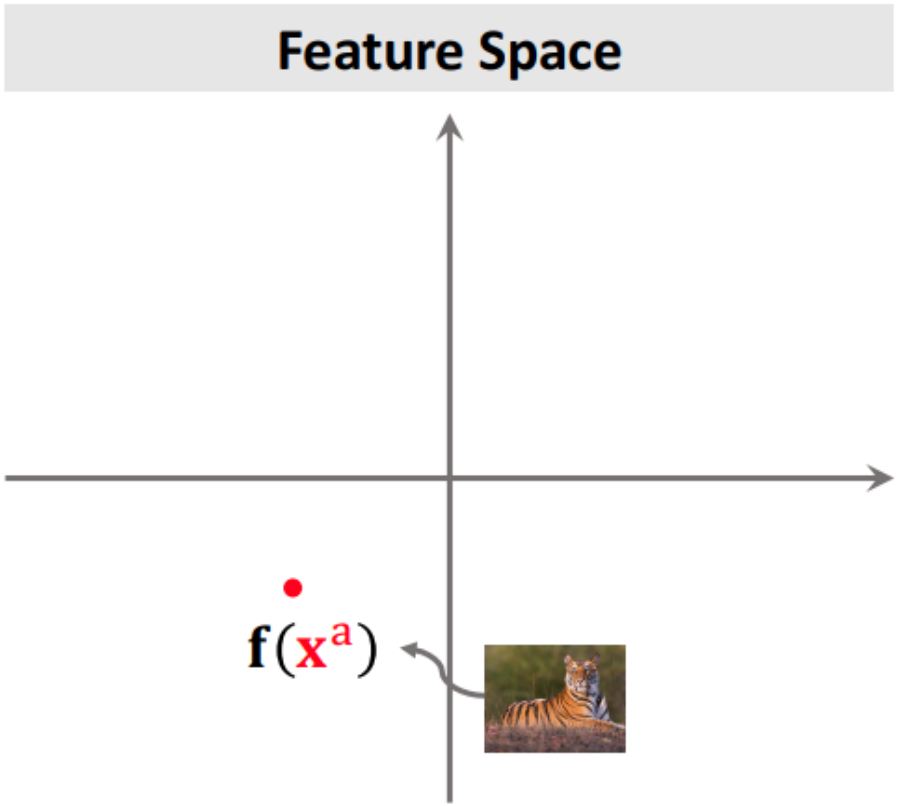


Triplet Loss

$\mathbf{x}^+$
(positive)

$\mathbf{x}^a$
(anchor)

$\mathbf{x}^-$
(negative)

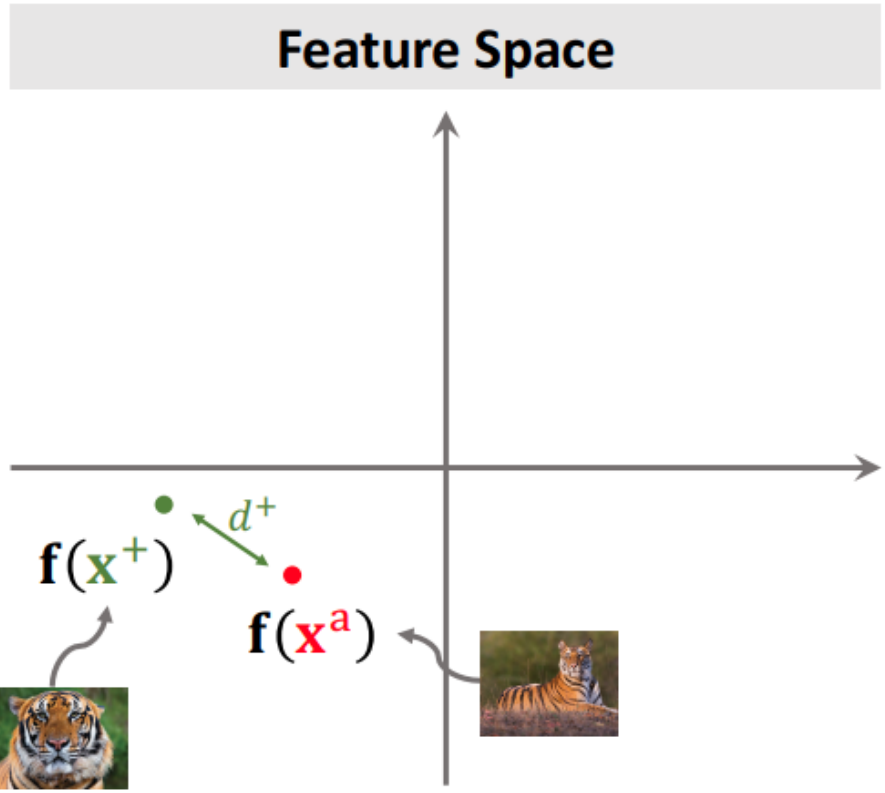


Triplet Loss

$\mathbf{x}^+$
(positive)

$\mathbf{x}^a$
(anchor)

$\mathbf{x}^-$
(negative)

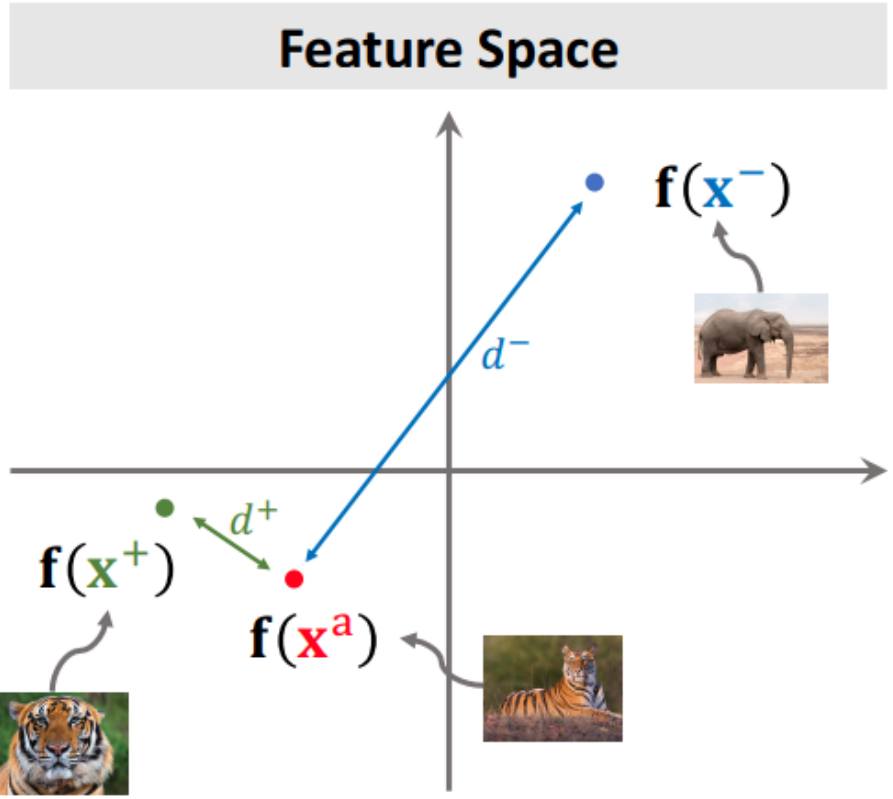


Triplet Loss

$\mathbf{x}^+$
(positive)

$\mathbf{x}^a$
(anchor)

$\mathbf{x}^-$
(negative)



Triplet Loss

$\mathbf{x}^+$
(positive)

$\mathbf{x}^a$
(anchor)

$\mathbf{x}^-$
(negative)

- Encourage $d^+ = \left|\left|\mathbf{f}(\mathbf{x}^+) - \mathbf{f}(\mathbf{x}^a)\right|\right|_2^2$ to be small.
- Encourage $d^- = \left|\left|\mathbf{f}(\mathbf{x}^a) - \mathbf{f}(\mathbf{x}^-)\right|\right|_2^2$ to be big.

Triplet Loss

$\mathbf{x}^+$
(positive)

$\mathbf{x}^a$
(anchor)

$\mathbf{x}^-$
(negative)

- Encourage $d^+ = \left|\left|\mathbf{f}(\mathbf{x}^+) - \mathbf{f}(\mathbf{x}^a)\right|\right|_2^2$ to be small.
- Encourage $d^- = \left|\left|\mathbf{f}(\mathbf{x}^a) - \mathbf{f}(\mathbf{x}^-)\right|\right|_2^2$ to be big.
- If $d^- \geq d^+ + \alpha$, then no loss. ($\alpha > 0$ is margin.)
- Otherwise, the loss is $d^+ + \alpha - d^-$.

Triplet Loss

$\mathbf{x}^+$ (positive)

$\mathbf{x}^a$ (anchor)

$\mathbf{x}^-$ (negative)

- Encourage $d^+ = \left|\left|\mathbf{f}(\mathbf{x}^+) - \mathbf{f}(\mathbf{x}^a)\right|\right|_2^2$ to be small.
- Encourage $d^- = \left|\left|\mathbf{f}(\mathbf{x}^a) - \mathbf{f}(\mathbf{x}^-)\right|\right|_2^2$ to be big.
- If $d^- \geq d^+ + \alpha$, then no loss. ($\alpha > 0$ is margin.)
- Otherwise, the loss is $d^+ + \alpha - d^-$.
- $\text{Loss}(\mathbf{x}^a, \mathbf{x}^+, \mathbf{x}^-) = \max\{0,\ d^+ + \alpha - d^-\}$.
- Update the CNN (function $\mathbf{f}$) to decrease the loss.

One-Shot Prediction

Query:

Support Set:

Fox

Squirrel

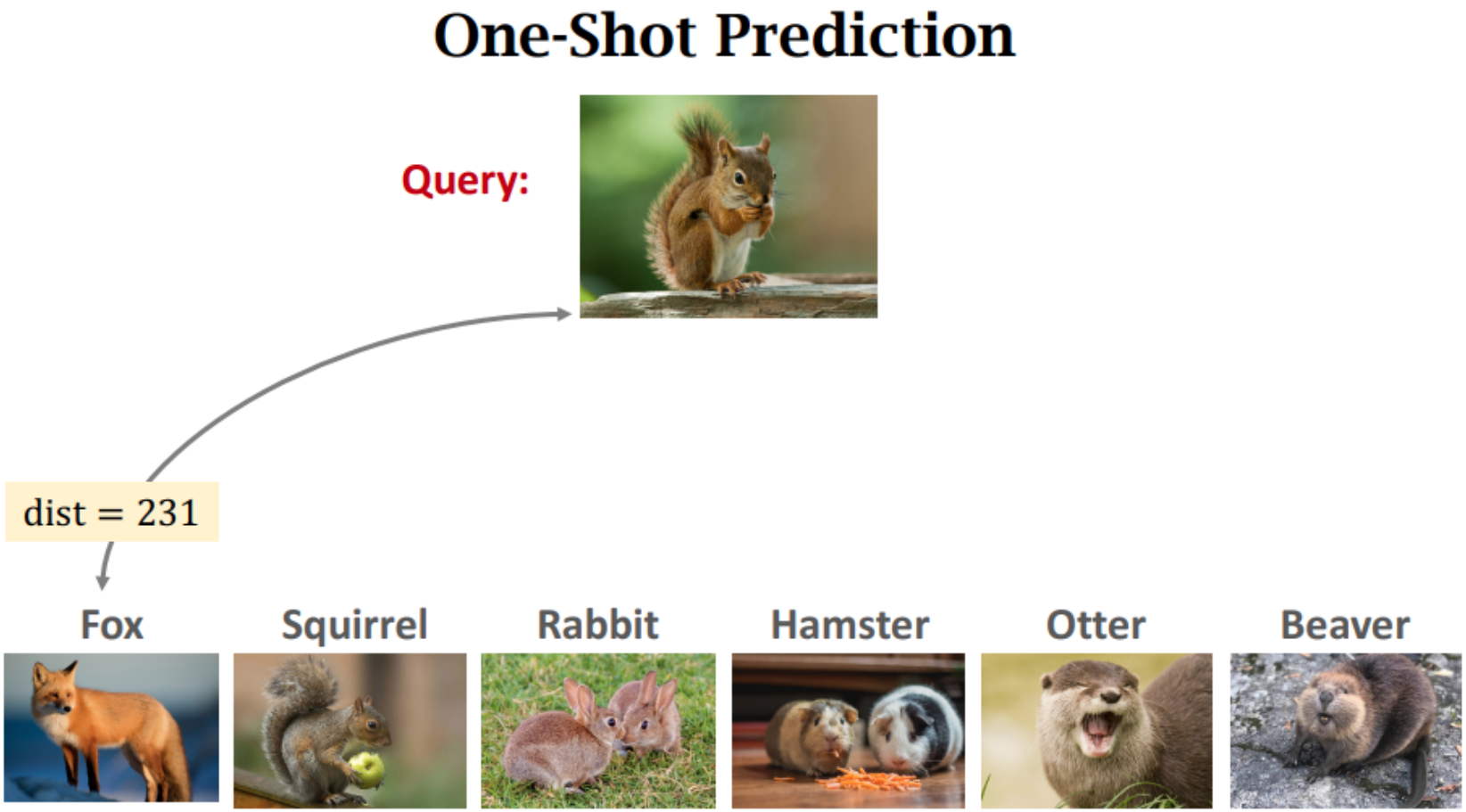
Rabbit

Hamster

Otter

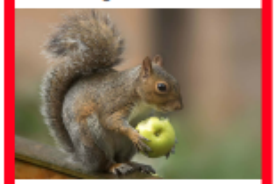
Beaver

One-Shot Prediction



One-Shot Prediction

Query:

dist = 231	dist = 19	dist = 138	dist = 76	dist = 122	dist = 94
Fox	Squirrel	Rabbit	Hamster	Otter	Beaver
					

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Zero-Shot and Meta Learning

Next Class